De Montfort University

Hugo Tassignon

Solutions to Non-Stationary Problems in Wavelet Space

PhD Thesis

School of Computing Sciences

Supervisor : Prof. Jonathan Blackledge

To my wife Myriam and my sons Lode and Tom.

Table of Contents

Table of Contents i

List of Symbols vii

Abstract viii

Acknowledgement x

Introduction : How the Thesis Developed and its Organisation xii

Chapter 1 : Literature Survey

1.1. Introduction p1
1.2. Signal Analysis before Wavelet era p1
1.3. Wavelets p4
1.4. Publication p8
1.4.1. Tutorial Papers p8
1.4.2. Applications p9
1.4. Books p10

Chapter 2 : System Theory

2.1. Introduction p12
2.2. Signals and Systems p12
2.3. Convolution p15
2.3.1. 1D Convolution p15
2.3.2. Convolution of Non Stationary Signals p17
2.3.3. 2D Convolution p17
2.4. Algebraic Modelling of the Convolution Operation p18
2.4.1. 1D Operation p18

2.4.2. 2D Operation p19
2.5. Transfer Function p20
2.6. Fourier Transforms p22
2.6.1. Fourier Integral p22
2.6.2. Fourier Series p23
2.6.3. Discrete Fourier Series p24
2.6.4. Discrete Fourier Transform (DFT) p25
2.6.5. 2D DFT p26
2.6.6. Discrete Cosine Transform (DCT) p27
2.6.7. Duality p28
2.7. Diagonalization of the Circulant Matrix p29
2.7.1. 1D Operation p29
2.7.2. 2D Operation p31
2.8. Convolution property p31
2.9. Modulation Property p32
2.10. Deconvolution p32
2.11. Laplace Transform p33
2.12. Z Transform p36
2.12.1. 1D Z Transform p36
2.12.2. 2D Z Transform p36
2.13. Chirp Z Transform p37
2.14. Non Stationary Applications p41
2.15. Conclusions p47

Chapter 3 : Algebraic Approach to Non-Stationary Convolution and Deconvolution.

3.1. Introduction p48
3.2. Non Stationary Convolution p49
3.3. The Convolution as an Algebraic Operation p51
3.4. Computer implementation p61

Chapter 4 : Digital Filters

4.1. Introduction p62
4.2. FIR Filter Design p63
4.2.1. Low Pass FIR Filter Design p63
4.2.2. High Pass FIR Filter Design p64
4.2.3. Band Pass FIR Filter Design p65
4.2.4. Band Reject FIR Filter Design p67
4.3. Windows p68
4.4. 2D FIR Filter Design p71
4.4.1. 2D FIR Low Pass Filter p71
4.4.2. 2D FIR High Pass Filter p76
4.5. Optimal Linear Filter Design p78
4.5.1. Introduction p78
4.5.2. Derivation of the Wiener Hopf Equations for 2D p79
4.5.3. Wiener Filtering in the Frequency Domain p82
4.5.4. Wiener filtering in wavelet space p84
4.5.5. Comparative 1D Study p84
4.5.6. 2D Wiener filtering example p85
4.6. Conclusions p103

Chapter 5 : Wavelet Transform

5.1. Introduction p104
5.2. Short Time Fouler Transform (STFT) p105
5.2.1. Resolution p106
5.2.2. Gabor Transform p108
5.3. Frames p110
5.3.1. Practical Considerations p113
5.4. Continuous Wavelet Transform (CWT) p114
5.4.1. From STFT to CWT : The Morlet Wavelet p114
5.4.2. Morlet Wavelet p116
5.4.3. Some Conclusions p120
5.4.4. Wavelets p121
5.4.5. Multiresolutional Analysis p125
5.4.6. Orthogonal Wavelets p128
5.4.7. Dyadic Sampling p132
5.4.8. Discretizing the Discrete Wavelet transform p132
5.4.9. The Haar Transform p135
5.4.10. Regularity p142
5.4.11. Daubechies Wavelets p145
5.4.12. Symmetry p152

Chapter 6 : 2D and Second Generation Wavelets

6.1. Introduction p153
6.2. The Two Dimensional Wavelet Transform p153
6.3. Biorthogonal Wavelets p161
6.4. The Lifting Scheme p163
6.4.1. Introduction p166
6.4.2. Split Step p167
6.4.3. Predict Step p168

6.4.4. Update Step....p170
6.4.5. One Level in the Wavelet Decomposition....p171
6.4.6. Other Prediction Functions : Interpolation Subdivision....p173
6.4.7. Interpolation in the Update Step....p176
6.4.8. Fast Lifted Wavelet Transform....p177
6.4.9. Implementation of the Fast Lifted Wavelet Transform with B-Spline filters p178
6.4.10. Example of an In-Place Calculation....p180
6.4.11. Software Realisations....p181
6.4.12. Conclusions....p182

Chapter 7 : Real Time Video Signal Processors

7.1. Introduction....p183
7.2. The Philips Video Signal Processor (VSP)....p184
7.2.1. Introduction....p185
7.2.2. Architecture of the VSP1 and the VSP2....p185
7.2.3. Programming of the VSP system....p190
7.2.4. Implementation of the Lifting Scheme....p195
7.2.5. Fixed Attenuation of the Subbands....p199
7.2.6. Wiener filtering....p205
7.2.7. Other methods....p211
7.2.8. Comparison and Conclusions....p211
7.3. The Texas Instrument TMS320C80....p213
7.3.1. Introduction....p213
7.3.2. Some Key Features....p213
7.3.3. System Architecture....p215
7.3.4. The TMS320C8X Software development board....p226
7.4. Wavelet Compression....p233
7.4.1. Definition....p233

7.4.2. Cost Functions p234
7.4.3. Thresholding p235
7.4.4. Variance p235
7.4.5. Entropy p235
7.4.6. Results p236
7.4.7. The Ripple-Effect p237
7.4.8. Parameters in Wavelet Compression p239
7.4.9. Wavelets vs. Wavelet Packets p239
7.4.10. Filters p240
7.4.11. Further Research p241
7.5. General Conclusions p242

Chapter 8 : General Conclusions p243

Mathcad's Short Reference Guide p246

Appendix 5a p251
Appendix 5b p269
Appendix 6a p273

References p279

List of Symbols

Symbol	Definition
$\mathbf{Z}$	Integers {....,-2,-1,0, 1, 2, 3....}
$\mathbf{N}$	Natural numbers { 0,1, 2, 3 ...}
$\Re$	Real numbers
l^2	Square summable sequences
L^2	Square-integrable functions
[a,b]	The closed interval $\{x : a \leq x \leq b\}$
[a,b)	The half-closed interval $\{x : a \leq x < b\}$
[a,b]	The half-open interval $\{x : a < x \leq b\}$
A^T	Transpose of the matrix A ; $A^T(m,n) = A(n,m)$.
$\langle u,v \rangle$	Inner product; $u,v \in l^2 \Rightarrow \sum_k u(k)v(k)$
$\|\| u \|\|$	Root-mean-square norm of u ; $\sqrt{\langle u,u \rangle}$
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
IFFT	Inverse Fast Fourier Transform
HDTV	High Definition Television
JPEG	Joint Picture Expert Group
MPEG	Motion Picture Expert Group
PCM	Pulse code Modulation
SNR	Signal To Noise Ratio
STFT	Short Time Fourier Transform
WT	Wavelet Transform
DWT	Discrete Wavelet Transform
FWT	Fast Wavelet Transform
MAC	Multiplier Accumulator
VSP	Video Signal Processor
MVP	Multimedia Video Processor
FIR	Finite Impulse Response
IIR	Infinite Impulse Response

Abstract

Non-stationary signals and systems are investigated in this thesis as to how they can efficiently be described, deconvolved, denoised, compressed and implemented on a fast processor by transforming the signal/system to another mathematical space. Shortcomings of classical Fourier transforms, in this field, can primarily be remedied by the introduction of time windows resulting in the Gabor transform. This proves not to be so satisfactory for our purpose. Finally, orthogonality of time and frequency windows are considered to effectively decompose the signal into a wavelet space. The associated discrete wavelet transform reshapes the sampled data of the signal into a more compact form : Specific properties of the signal are not only better understood but also more compactly represented in the new space .Deblurring for instance, becomes more effectual. This is crucial when investigating real-time processing. An efficient algorithm not only tells us how the signal processing could be quickly implemented, it can also reveal what hardware looks most appropriate for the analysis.

In this context it is interesting to compare FIR filtering with wavelet filtering : They both use convolution, the wavelet filter is, leaving decimation/interpolation out of consideration, a very specific FIR filter with a natural finite support and very much governed by orthogonality conditions. The classical FIR filter always makes use of a truncated (windowed) set of filter coefficients and judges a 'brickwall' frequency characteristic as the primary objective. For the hardware we conclude that there is a common need on intensive Multiply-Accumulate (MAC) operations.

With the appropriate choice of wavelet transform, the wavelet space contains less correlated data . One could see this as a lossless compression operation. This allows statistical filters like Wiener filters to be more efficiently run.

Considering implementation, two problems arise. First, how fast can a discrete wavelet transform be run on a computer. Biorthogonal transforms with the lifting scheme are the ultimate choice.(See Chapter 6). Second, on what microcomputer chip can these algorithms best be implemented.(See Chapter 7). Digital signal processing requires a lot of parallelism to boost the performance to a real-time processing level. Two systems are investigated : the VSP system from Philips with a vast amount of

primitive ALUs and the MVP from Texas Instruments with one master and four elaborated slave processors. The last one proved to be the most effective one. When many processors are involved in a real-time process it turns out to be extremely difficult to equally divide the weight of the algorithm over the minimal amount of processors.

Chapter 8 comprises a summary of the main results and suggestions for further research.

Acknowledgement

First of all I would like to thank my wife Myriam for her patience, understanding and unconditional support during all the years that I have worked on my PhD. There were probably more agreeable subjects than to hear holidays and other social activities being cancelled or postponed because of professional and PhD priorities. I promise here to make it up to her and to pursue a better balance between professional and leisure activities.

To Prof. Blackledge, for his open mind and his non-conformist approach to scientific problems and the world in general. He immediately took interest in me when I visited Cranfield University and was very flexible and understanding when I made suggestions to redirect my PhD to the area of wavelets. He was always very interested in my work and very co-operative in establishing new projects. I would like to thank him and his wife Hannah as well for their kind hospitality in Leicester.

To the staff of De Montfort University, who have allowed me to continue my PhD after one year at Cranfield University.

To my friend and colleague Ing. Fons Sinnaeve who encouraged me to start a PhD. He was my most loyal supporter during the good and the bad days. His suggestions as an experienced writer and expert in biomedical engineering were always very valuable to me.

To Dr. Rousseau, mathematician, bibliometrian and perfectionist in writing mathematics. He not only sharpened my mind to purify my mathematical language, he also provided me with an extensive amount of publications on wavelets. This was the basis for a large part of my PhD work and it allowed me to concentrate on the applications of wavelets.

To the direction of my Institute, the Katholieke Hogeschool Brugge Oostende (KHBO) : To Dir. Gerard Manderyck for his empathy for all European programs including my PhD. To Prof. Egide Degroote, Dean of the Department of Industrial Sciences and Technology, for his positive and flexible attitude concerning all problems related to my PhD.

To all my colleagues who supported me in this work with suggestions, projects or just sympathy : Prof. Jef Vanneuville , lic. Guido Herweyers , ir. Mark Verstraete , Prof. Johan Catrysse , Ing Noel Lagast, Ing Paul Lapere... and many others from the Institute where I work.

To my MSc. students Francis Decroos, Kris Dierkens, Vincent Pollet and Koen Verweyen , for their close collaboration, help in writing, programming and doing experiments with wavelets.

Last but not least I would like to thank Marina and Albert Markham, in Stevington near Bedford . The world is full of surprises : It was my wife Myriam who first met Marina Markham at a bilateral meeting on lace some 10 years ago in Bedford. When we became acquainted with the Markhams, I was not aware of Cranfield University at that moment. It was only when I thought of visiting it, about 5 years ago, and when I looked it up on the map that I was astonished how close Cranfield was to Stevington. Their kind hospitality helped me to start this PhD. I am very grateful to them for letting me stay with them during my visits to England.

To conclude, I would like to show my gratitude to all the scientists who not only enhanced the scientific knowledge but also for their personal opinion in which they expressed on how to get on with it. In particular there is Alan V. Oppenheim whose book on 'Digital Signal Processing' is the standard work . I bought it in 1980 and it gave me great satisfaction in reading it and becoming aware of this fascinating discipline. In 1992 he wrote a very interesting article 'A Personal View on Education' [44] in which he expressed how teachers should be mentors and friends to their students. He also emphasised the role reversal where teachers should become students. He is a great scientist and a great man.

To my last and final mentor, I would like to quote from Albert Einstein. I tried to keep it in mind while writing :

'Everything should be made as simple as possible, but not simpler.'

Introduction : How the Thesis Developed and its Organisation

In 1990, my Institution established a number of postgraduate collaborations with English, French, Czech and Slovak Universities in the field of avionics and micro electronics. The activities with Western European Universities concentrated on curriculum and course development for the STAR (Specialised Training in Aeronautics and Research) program. In collaboration with the Technical Universities of Prague , Brno and Bratislava in Eastern Europe we organised an MSc course and a research program in micro electronics based on a TEMPUS project which involved me as a lecturing in digital signal processing (DSP).

It was with this background that, early in 1993, I contacted Cranfield University with the view of collaboration in avionics. As I was a specialist in DSP it seemed an intriguing idea to me to further develop my DSP knowledge in the field of aerospace techniques. A PhD was a good synthesis of co-operation in research in a well established European University .

I was very pleased when I first came to Cranfield, that Prof. Jonathan Blackledge wanted to consider my vague proposal for collaboration. He introduced me to one of his MSc students, Patrick Lezeau, who had finished his thesis on 'Solutions to the 2D Non-Stationary Deconvolution Problem' . He asked me if I was interested in non-stationary problems and I said yes although I only vaguely knew what he meant because until then , as an engineer, I had almost exclusively been busy with stationary problems . After reading Lezeau's thesis I became very much interested in the subject and I made an agreement with Prof. Blackledge to start a general study on non-stationary problems in DSP. That is where it started!

Being a part time student it was important to me that there were some common interest between the activities at the school of Computing Sciences and my Institution, the KHBO. Therefore the idea was developed to situate the thesis as follows :

- At the DMU : Investigate and develop new algorithms for non-stationary signals and systems.

- At the KHBO : Implement these algorithms on fast video processors. Investigate the performance and develop new visions for future semi-custom and full custom DSP and image processing chips.

In the first year I investigated the behaviour of stochastical processes to become aware of non-stationary problems and their possible solutions. I also studied image processing to become conscious of the complexity of 2D problems. My first temptation to tackle non-stationary problems was by exploring the field of stationary techniques and the way in which way they were applicable.

The use of the chirp z transform was a first attempt to detect a stationary signal after it had been convolved with a non-stationary kernel. The results are described at the end of **Chapter 2**. This chapter contains also some fundamentals on Signal Theory. Although not strictly being part of the non-stationary problem, I found it useful to incorporate it in a general review on Signals and Systems. It reminded me of linearizing techniques in non-linear network analysis. When signals in an electronic analogue system become larger, the response of the system can gradually alter becoming non-linear. An interesting field of research concerns the weakly non-linear systems where both system descriptions meet. Examinations of non-linear systems often make use of linear techniques as a first approximation to the solution. The same is true for stationary / non-stationary signals and systems.

Chapter 3 considers the numerical techniques for non-stationary deconvolution in 1D and 2D. Considering the nature of speech processing and knowing the Linear Predictive Coding (LPC) technique I became aware of the need to analyse non-stationary signals in such small time or space frames that they are by themselves only weakly non-stationary (i.e. approximately stationary). This brought me to Short Time Fourier Transforms and later on to Gabor and Wavelet Transforms.

I had only been involved with Fourier , Laplace and z Transforms in filter and control engineering. It seemed interesting to me to make an historical and a technical study of how these transforms were discovered and how they gradually became mature disciplines.

I started with Joseph Fourier and his harmonic decomposition of periodic signals and ended with Luc Sweldens lifting scheme for fast biorthogonal wavelet

transforms. This evolution is described historically in Chapter 1 where it is presented as a literature survey.

Chapter 4 not only treats classical digital filters in 1D and 2D, it also concentrates on Wiener filters and how they can be used in wavelet spaces to remove noise out of a noisy signal . Wavelets are used before they are explained. This was done to keep all the Wiener filter applications together. We refer to Chapter 5 for all theoretical background on wavelets.

Chapter 5 depicts in a mathematical language the evolution from Fourier analysis to wavelet transforms. It ends with some applications of Daubechies' wavelets.

When I started my research there were few publications on wavelets and on how to build them. I spend quite some time in my second year on designing some proper wavelets and the software to do the analysis. I found it very inspiring and it gave me a profound insight in to what wavelets are and how they should be used. This is reported in Appendix to the Chapters 5 and 6.

It was initially frustrating when I first started surfing on Internet and I found many reports on applications I was working on. The wavelet construction was one of them. Thus, in my third year, which was almost completely oriented to applications like real-time Wiener filtering in wavelet space. I consulted the Internet intensively and reoriented my research to algorithms, which were reported on the Internet but not yet officially published like for example Sweldens' lifting scheme for biorthogonal wavelets. Consequently it was implemented in a Video Signal Processor (VSP), which was my original contribution to the field. The Internet can't be ignored in modern research, it is a powerful window on science if one can risk looking at it. It very much broadens the view on how the world responds on new techniques and it helps one to keep the pace with new developments.

Chapter 6 looks more deeply at the wavelet problem and explores new wavelets such as biorthogonal wavelets and algorithms such as the lifting scheme for example which is not based on Fourier transforms . This lifting scheme is thoroughly investigated on its efficiency for further applications on VSPs.

In **Chapter 7** new processors for real time image processing such as the Philips Video Signal Processor (VSP) and the TMS320C80 from Texas Instruments are explored. Non-stationary signal processing like noise reduction and image compression are implemented on them. Finally some philosophical thinking is produced how ASICs for real time image processing could be build efficiently.

Integrated in the text, some Mathcad® files were introduced. Although these instructions are mathematically well edited and understandable, a review of the most important Mathcad instructions is made at the end of the thesis.

Chapter 8 formulates the results of the thesis and makes some suggestions for further research.

To conclude, my PhD work, reported in this thesis , produced a general review on stationary and on non-stationary problems in digital signal and image processing. It was not the intention to focus on one particular problem but to look at the area in the broadest way and to come to algorithms which could efficiently be implemented on VSP chips. The final goal was to find out which computer architecture was most appropriate for the treatment of non-stationary signals.

For me, personally, it was a fantastic experience and it enhanced very much my mathematical knowledge in the field of DSP. As an engineer, it gave me the opportunity to look at applications and implementations with a much greater palette of tools then I had before. However, to quote Leonardo da Vinci : 'There is no higher or lower knowledge, but one only, flowing out of experimentation'.

I am sure that ,in the future, this acquired knowledge will result in a successful collaboration between my Institute and the Department of Mathematical Sciences of the De Montfort University. An ESPRIT proposal is already established where the techniques described will be used.

CHAPTER 1

Literature Survey

1.1. Introduction

The intention of this literature survey is first to present a brief overview of the evolution of signal analysis during the history of mathematics. Yves Meyer gives in his book ' Wavelets, Algorithms & Applications' , an interesting historical perspective on Wavelets. [61] . Very recently a new book came on the market which tells the story of wavelets 'The World According to Wavelets' which depicts in a very accessible way wavelets , how they were created and the people involved with the topic.[58]

In this Chapter a review of the consulted literature, especially on wavelets, is given. A systematic discussion on what wavelets are , how they are constructed and where they can be applied, is given.

Finally, books and literature on other subjects such as non-stationary deconvolution, chirp z transform , Wiener filtering are reviewed .

1.2. Signal Analysis before Wavelet Era

Returning to the origins brings us to Joseph Fourier. He asserted in 1807 that any 2π-periodic function x(t) can be reconstructed with a sum of sines and cosines :

$$x(t) = a_0 + \sum_{k=1}^{\infty} a_k \cos(kt) + \sum_{k=1}^{\infty} b_k \sin(kt) \qquad (1.1)$$

with the well known expressions for a_0, a_k, b_k. (See Chapter 2)

These were very surprising results and they focused mathematicians on a better understanding of functions and integrals. In fact until then entire series were used to represent and manipulate functions. By passing from power series like :

$$a_0 + a_1 t + a_2 t^2 + a_3 t^3 \ldots \tag{1.2}$$

to expressions of the form

$$a_0 + \sum_{k=1}^{\infty} a_k \cos(kt) + \sum_{k=1}^{\infty} b_k \sin(kt) \tag{1.3}$$

a whole new world of mathematical thinking was introduced.

In 1873, Paul Du Bois-Reymond constructed a continuous, 2π-periodic function of a real variable , whose Fourier series diverged at a given point. For Fourier's statements to stay true some refinements on function notation, convergence of Fourier series and **research on other orthogonal systems** were necessary.

Lebesgue formulated the functional concept best suited for Fourier series. It involved the space $L^2[0,2\pi]$ of square integrable functions on the interval $[0,2\pi]$.

Another research path led to wavelets. This path was followed by Haar who wondered if there existed orthogonal systems other than those based on sines and cosines.

In particular could a set of orthogonal functions $\{h_0(t),h_1(t),\ldots,h_n(t),\ldots\}$ defined on $[0,1)$ be found so that for any function $x(t)$ continuous on $[0,1)$ the series

$$\langle x,h_0\rangle\, h_0(t) + \langle x,h_1\rangle\, h_1(t) + \ldots + \langle x,h_n\rangle\, h_n(t) + \ldots. \tag{1.4a}$$

$$\text{where } \langle x,h_n\rangle = \int_0^1 x(t)h_n(t)dt \tag{1.4b}$$

converges to $x(t)$ uniformly on $[0,1)$?

In 1909 Haar discovered a very simple solution, not imagining that he in fact opened the route to the discovery of an infinite number of solutions . The Haar system makes use of a **non continuous block function** .

The basic function is $h_1(t) = 1$ on $[0,½)$,-1 on $[½,1)$, and 0 outside the interval $[0,1)$ .Translated and dilated versions of $h_1(t)$ form an orthogonal set. (The theory and applications of this is described in more detail in Chapter 5) . One criticism on the Haar construction is that the building blocks $h_n(t)$ used to construct continuous functions are themselves not continuous functions ! The approximation with a limited number of terms in Eq. (1.4) results in a **sample and hold** action with the mean value of x(t) during the hold time being a sample value. Another problem arises when continuous functions with continuous derivatives are to be approximated by a Haar system. This situation is completely inappropriate and leads to an infinite number of terms! A much better solution in this case would be a polynomial or higher order approximation instead of the simple sample and hold rule.

Faber and Schauder replaced the block function by a triangular one with basic function : $\Delta(t) = 2t$ on $[0,½)$, $2(1-t)$ on $[½,1)$, and 0 outside the interval $[0,1)$. Once again a set of functions could be constructed with dilated versions of the basic function and a so called Schauder basis could be constructed.

In 1927, Philip Franklin a MIT professor had the idea of building an orthogonal basis starting from the Schauder basis. It had the advantage of decomposing any function in $L^2[0,1)$ which the Schauder or Haar bases could not. The weakness of the Franklin basis was that it had no longer a simple algorithmic structure. The functions in a Franklin basis are unlike those of the Haar or Schauder basis; they are not derived from a fixed function by integer translations and dyadic dilations. This defect caused the Franklin system to be abandoned for almost 40 years. It was Stromberg who discovered in 1980 an asymptotic estimate of the Franklin function which could be translated and dilated to construct an orthogonal basis.

Before coming to recent times it is important to mention that in the 1930s scientists were interested in mathematical tools to describe the Brownian motion and hence random signals. This brings us to **non-stationary signals**. A straightforward procedure, with the Fourier transform in mind, was to analyse these signals in time

slots . A time translatable window was created to slide over the signal. The Short Time Fourier Transform (STFT) was born. In 1946 Dennis Gabor [40], well aware of short time changing frequencies in speech and music, wrote that : In the framework of Fourier analysis where sines and cosines **oscillate for all times,** it would be a contradiction in terms to consider short time changing frequencies. He soon became aware of a painful compromise : The smaller the window the better sudden changes could be located ,but the blinder one becomes to lower frequency components in a signal. By choosing a bigger window, more could be seen of the low frequencies but at the expense of localising a phenomenon in time . Gabor was one of the first to relate the Heisenberg uncertainty principle in physics to signal and communication theory . So he introduced the principles of time resolution (Δt) and frequency resolution (Δf) into signal analysis. He also found a lower bound solution for the Heisenberg uncertainty principle , i.e.

$$\Delta t \, \Delta f \geq 1/4\pi \qquad (1.5)$$

The Gaussian window seemed to be the lower bound solution for the Heisenberg compromise. Nonetheless Gabor had to admit that with his 'Gabor Transform' it was not possible to create orthogonality as with the classical Fourier Transform.

1.3. Wavelets

Meyer[61] states that it is almost a job for an archaeologist to find the roots of wavelets! Wavelets were implicit in mathematics, physics, signal and image processing, and numerical analysis long before they were given the status of a unified scientific discipline. As a starting point however he takes the work of Jean Morlet, a geophysicist working for the French oil company Elf-Acquitaine who developed wavelets as a tool for oil prospecting.

The standard way to look for oil , introduced in the 1960s, is to send seismic impulses underground and to analyse their echoes. This analysis is supposed to tell how deep and how thick the various layers are and what they are made of. Roughly

speaking, the frequencies of the echoes are linked to the thickness of the various geological layers, with high frequencies corresponding to thin layers.

Around 1975 Morlet reintroduced the 30 years old Gabor Transform but ran into the impreciseness of time measurements in the high frequencies. Another serious drawback was the absence of a numerical algorithm to reconstruct the signal from the transform.

Thus Morlet took another approach. Instead of keeping the window fixed and filling it up with oscillations of different frequencies he did the opposite ; he kept the number of oscillations in the window constant and varied the width of the window by stretching or compressing it. He called them **wavelets of constant shape** to distinguish them from what he called the Gabor wavelets.

Morlet developed empirical methods for decomposing signals into wavelets and reconstructing them. His work was rejected by colleagues and he sought a stronger mathematical support in Marseille where Alex Grossmann was working on quantum mechanics and signal processing.

There was the 'Morlet recipe' for the transform but it was not clear whether its numerical tools were true in general or just approximations . To validate the empirical work, Grossmann and Morlet [19] first showed that the energy of the signal was unchanged under transform action so that one could exactly transform back and forth. Unlike the Fourier transform where only **one integral** was involved in the **inverse** transform this Morlet transform needed a **double integral** due to the double variable (time and frequency) which was involved. Finally,[19] they found a good approximation with one integral but then started worrying about the error. It was only after many numerical experiments that they dared to say that the error was zero!

Thus the first description of a wavelet was produced. The next step was to consider orthogonality. In 1981 Roger Balian thought he could prove that it was not possible to have an orthogonal representation with Gabor transform. Meyer had a similar idea and thought he could prove that orthogonal wavelets didn't exist. In 1985 he failed, by precisely constructing by trial and error, the kind of wavelet he thought didn't exist! Daubechies [57] wrote later that it was almost a miraculous discovery, because it was realised without an underlying concept as Meyer later admitted.

Another disappointment for Meyer was to find that Strömberg had created another orthogonal wavelets 4 years earlier.

This was really the start of an increase in consciousness about different research fields all heading in one direction. More progress was made when in 1986 a 23-year-old student, Stephane Mallat said to Meyer that the multiresolutional analysis as wavelets was essentially the same thing that electrical engineers had been doing under other names. Meyer stated that : This was a complete new idea, the mathematicians were in their corner, the electrical engineers were in theirs, the people in vision research like David Marr were in yet another corner, and the fact that a young man like Mallat was capable of saying that they were all doing the same thing and that they had to look at it in broader perspective , that was quite remarkable!

A publication [9] resulted out of this meeting in which it was made clear that the work that existed in many guises and under different names like wavelets, the pyramid algorithms used in image processing, the subband coding of signal processing, the quadrature mirror filters of digital speech processing were at heart all the same. Using wavelets to look at a signal at different resolutions can be seen as applying a succession of filters, filtering out high frequencies with short wavelets and low frequencies with long wavelets.

Wavelet theory benefited very much from Mallat's original ideas. For example, the new concept of a **scaling function** made it indirectly possible to construct a **Fast Wavelet Transform (FWT)** . Ronald Coifman said that Mallat's fast computing algorithm had the same effect on non-stationary analysis as John Tukey's FFT algorithm on classical Fourier analysis. Mallat also described in a framework how to construct new orthogonal wavelets.

Unlike the Morlet or Meyer wavelets these new wavelets were created using iteration processes. Meyer said of this : Mallat launches brilliant ideas that keep two or three hundred people busy, then he goes on to something else. It was Ingrid Daubechies with her tenacity, her capacity of work, who implemented it.

Daubechies wanted to construct wavelets for which it took much less computational time to calculate coefficients than Meyer's infinite orthogonal wavelets . Daubechies' constraints were very severe ; in addition to orthogonality she

wanted the wavelets to be finite (i.e. with compact support), very smooth and with vanishing moments. (See Appendix 5B) . She could create a whole set of wavelets , starting with the Daubechies2 (2 filter coefficients) which was found identical to the Haar transform and proceeding with Daubechies4,...Daubechies20 . The technique is explained in Appendix 5b.

When the euphoria of all this splendid mathematical theory past, researchers started asking questions about how to apply it efficiently in for example speech and image processing where new ideas started to be voiced .

The biorthogonal wavelet was created to allow the use of symmetrical filter coefficients . Orthogonal wavelets were all constructed with asymmetrical filter coefficients.

Wavelet packets were constructed by making use of the best basis technique. Instead of finishing with the wavelet coefficients based on Mallat's scheme (for bandpass filtered signals) all spaces are continuously split up in two equal subspaces., Information costs are calculated for the original space and the two sets of coefficients in the subspaces. Information costs in all spaces are compared and minimal values are used as a directive to construct the best basis .(See Chapter 7) This gives better compression ratios than with plain wavelet analysis for example.

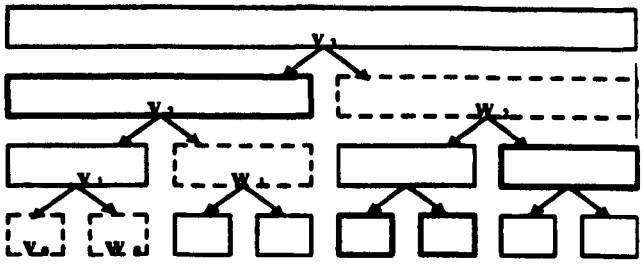


Fig.1.1. Schematic representation of a space and its subspaces after using splitting on 3 levels. The top rectangle represents the space V3 and each other rectangle corresponds to a certain subspace of V3 generated by wavelet packets. The slanted lines between the rectangles indicate the splitting. The dashed rectangles then correspond to the wavelet multiresolutional analysis $V_3 = V_0 \oplus W_0 \oplus W_1 \oplus W_2$ *(See Paragraph 5.4.5). The bold rectangles correspond to a possible wavelet packet splitting which is, dependent on the information cost, a possible best basis.*

The lifting scheme could be seen as a kind of turbo FWT . By using biorthogonal wavelets and taking into account the symmetry in its filtercoefficients Sweldens et al. [12] were able to construct a very efficient algorithm almost at the speed of FFT! This is used in the research on noise reduction using Video Signal Processors (VSP) in this thesis.

New fields of research are wavelets for irregular sampled data where the lifting scheme could be very useful. Also wavelets on spheres are under investigation. They could for instance be used in space research on the characteristics of the earth's and other planet's surfaces. These and other new topics were suggested by Wim Sweldens [18]

This historical review is not intended to be complete. In fact, I used it as a guideline for my study of wavelets. It gave me a better understanding how it could be embedded. In Chapter 5 further mathematical details are presented.

1.4. Publications

1.4.1 Tutorial Literature

The first paper I was confronted with when starting my Ph.D. work was the article of J.M. Blackledge and P. Lezeau [1] which was a compilation of Lezeau's MSc thesis at Cranfield University. It investigates the algebraic solutions of non-stationary deconvolution.

Seeking for alternative techniques for non-stationary signal analysis I became interested in wavelets. The fist article I found enticing was an IEEE paper of O. Rioul and M. Vetterly [2] : It developed a good balance between historical evolution of signal analysis and mathematical background, written from an engineers viewpoint.

This definitively put me on the wavelet trail and articles like Bultheel 's [4] one affirmed to be a very comprehensive introductory article. An explanation of the mathematical background was found in [5]. Once the basics understood I went

searching for implementation algorithms : R. Strichartz [7] , C. Taswell and K. McGill [8] describes on how to implement Mallat's Fast WT in an algorithm.

I also explored the origin of wavelets this brought me to some basic articles from Grossmann and Morlet [19] and Daubechies [13],[17]. In Hubbard's book [58] a lot of eminent mathematicians testify about their experiments with wavelets; very interesting to read and inspiring for a better understanding of wavelets.

With some algorithms, written in Matlab, the need for better understanding of Mallat's scheme brought me to his article on multiresolutional analysis [9]. At that time I started searching on Internet for new visions on wavelets . So I found W. Sweldens' publication on multiresolutional analysis.[11] and its presentation of the lifting scheme [10] . This new idea opened new interesting prospects on wavelets : The use of spline polynomials for the construction of biorthogonal wavelets in a fast lifting scheme with simple arithmetic.

The study of Unser's articles [20], [21] certified my vision on the importance of spline polynomials for the construction of biorthogonal wavelets.

As a last interesting article I found Sweldens' [18] : 'Wavelets : What Next?': Very inspiring.

1.4.2. Applications

Most of my work was oriented to deconvolution in the very broad sense : non-stationary deconvolution, noise reduction, object identification, etc. The papers consulted were either directly or indirectly used in the applications reported in this thesis. There is for instance information about spline functions implemented in the lifting scheme for biorthogonal wavelets.[12] There are also publications from astrophysicists trying to make better pictures of the universe .[24],[25],[26],[27],[28],[29] and there are papers from people working in Morlet's footsteps and attempting seismic exploration to extract hidden information by using Gabor and Wavelet transforms [30],[31].

In this context, two-dimensions is a natural next step . Image processing , much more then one-dimensional signal processing, needs efficient algorithms . Applications such as direction detection were studied.[34],[35],[36].

The IEEE Engineering in Medicine and Biology [33] looks at a number of uses of Wavelets in Image Processing. All kinds of biomedical signals are analysed.

Transforms performed on data normally result in a more compact organisation of the data. Neural networks can profit from this because they need less data for training. [60]

There is currently considerable efforts being invested to extract music, originally performed by Johannes Brahms, from an old wax drum .This is reported in[39] .

1.5. Books

Ingrid Daubechies' 'Ten Lectures on Wavelets' [57] proved to be the most valuable book on wavelets. She was a privileged witness at the birth of wavelets. Next to the mathematical explanation she gives a lot of interesting background comments about the why and how. Considering the time when it was written, in my opinion not yet surpassed.

Yves Meyer's book :' Wavelets , Algorithms & Applications' [61] is based on lecture notes written by Meyer. He is a brilliant mathematician says Daubechies, an important personality in the wavelets but didactically this book is not so well built. He present an interesting historical review on wavelets but considering it appeared a year later on the market than Daubechies' book , it is not really an original publication.

B.Ruskai (ed) 's 'Wavelets and their applications ' [65] is just a collection of articles with a broad field on applications.

Gilbert G. Walter 's 'Wavelets and Other Orthogonal Systems with Application.' is a rather disappointing book. The publisher presents it with the comment "Unlike most other books that are excessively technical, this text presents the basic concepts and examples in a readable way". There is indeed less mathematics in it than in Daubechies' book but it does not make it easier to understand it because, at least in

my opinion, the author by omitting some fundamental mathematics, produces sometimes sloppy explanations or proofs .

M. Victor Wickerhausen's 'Adapted Wavelet analysis from Theory to Practice' is, referring to the publisher " a detail-oriented text , intended for engineers and applied mathematicians who must write computer programs to perform wavelet and related analysis on real data". It provided me with a lot of information and almost ready to use C software for wavelet analysis and synthesis.

CHAPTER 2

System Theory

2.1. Introduction

In the preceding chapter a historical review was first made of the evolution in signal and system theory, the field to be researched was explored and a scenario for research was outlined.

Because a lot of advanced mathematics will be involved in non-stationary signal analysis a sound foundation on basic system theory is desirable. In this Chapter sampling, convolution, deconvolution and Fourier transforms are defined together with the z-transform and the limitations when confronted with non stationary signals. Some related research field such as the chirp z transform are also discussed.

2.2. Signals and Systems

We will consider analogue and discrete signals and systems. Mathematically a treatment of signals and systems is identical. Physically however, they have a totally different meaning . Signals will be used to examine systems or as mathematical expressions on how systems respond to stimulating signals. We will use expressions like f(t),s(t), x(t), y(t), ... for analog signal and reserve h(t) for system notation, (t : continuous time notation; $t \in \Re$).The discrete versions will be $f(n.t_s)$, $s(n.t_s)$, $x(n.t_s)$, $y(n.t_s)$,.. $h(n.t_s)$ $n \in \mathbf{Z}$ where t_s is the sampling time and f_s the sampling frequency = $1/t_s$). The shorthand notation will be f(n), s(n), x(n), y(n),.. h(n).

In two-dimensions we will use expressions like f(x,y), g(x,y), s(x,y) for images. (x,y will in this case stand for a continuous space notation in the plane; $x,y \in \Re$).

Systems like filters for instance acting upon images will use the expression h(x,y). Note that these continuous expressions have only a theoretical meaning but could be advantageously used in transforms to mathematically study the subject. Since digitisation is possible, image processing makes use of discrete co-ordinates. The pixel distance is never mentioned; it is in fact dependent on the size of screens and paper on which the image is 'projected' . The discrete versions will be notated as $f(n_1,n_2)$ $g(n_1,n_2)$, $s(n_1,n_2)$, $h(n_1,n_2)$. $(n_1,n_2 \in \mathbf{Z})$. If necessary, pixel distance can be calculated by inverting the pixels per inch expression (ppi).

We will distinguish 3 kinds of analog and discrete signals and systems :

1. Specific test signals :

   - $\delta(t)$, $\delta(n)$: one dimensional delta pulse. Note that $\delta(t)$ is an impulse of infinite height with $\int_{-\infty}^{\infty}\delta(t)dt = 1$. (Theoretically useful but not practically manageable). $\delta(n)$ is just a pulse of amplitude 1 at n=0.
   - $\delta(x,y)$, $\delta(n_1,n_2)$: two dimensional delta pulse.
   - $u(t)$, $u(n)$: one dimensional unit step.
   - $u(x,y)$, $u(n_1,n_2)$: two dimensional unit step.
   - $u(t) - u(t-\tau_0)$, $u(n) - u(n-m)$: one dimensional pulse of width τ_0 (analog) , m (discrete).
   - $u(x,y) - u(x - x_0 , y - y_0)$, $u(n_1,n_2) - u(n_1-m_1,n_2-m_2)$: two dimensional analog and discrete pulses.

2. Deterministic and stationary /nonstationary signals and systems. We distinguish periodic and non-periodic signals and systems :

   - $\sum_{m=-\infty}^{\infty}\delta(t-mt_s)$; $m \in \mathbf{Z}$ or discrete $\sum_{k=-\infty}^{\infty}\delta(n-k)$; $n,k \in \mathbf{Z}$: Analog and discrete delta pulses series.

- $\sin(2\pi ft)$, $\sin(2\pi n.f/f_s)$ or also $\sin(\omega t)$, $\sin(\Omega.n)$ with $\omega=2\pi f$ and $\Omega=2\pi f/f_s$
- $e^{j2\pi.f.t}$ $e^{j2\pi.n.f./f_s}$ or also $e^{j.\omega.t}$, $e^{j.\Omega.n}$. (periodic)
- $e^{j.\Omega_1 n1}$ $e^{j.\Omega_2 n2}$: Ω_1 and Ω_2 are in this case the two-dimensional frequency (angular velocities) variables.
- nonstationary signals : discontinuous steps in amplitude response of sinusoidal functions result in nonstationary behaviour .

  An example : $x(t) = \{ \sin(\omega_1 t)$ for $0 \le t \le \tau_1 + \sin(\omega_2 t)$ for $\tau_1 < t \le \tau_2 \}$.
- specific non-periodic signals are very often impulse responses of systems described by exponentially decaying functions. This property is very important because it will make the integration or summation (eq. 2.3) possible.

A condition for this will be : $\int_{-\infty}^{\infty} h(t)dt < \infty$

or in discrete form : $\sum_{n=-\infty}^{\infty} h(n) < \infty$

Some examples : $h(t) = e^{-at}$ $(a>0)$ or discrete $h[n] = \alpha^n.\sin(\beta.n)$ $(0<\alpha<1)$

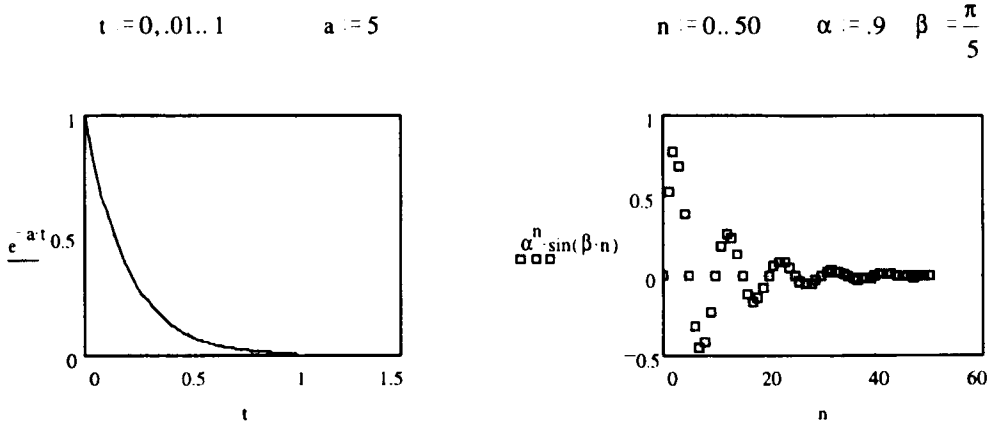


Fig.2.1. Analogue and discrete impulse responses.

3. Non-deterministic stationary/non stationary signals.

Many of the one or two-dimensional signals we want to examine like f(n) : sampled speech or $x(n_1,n_2)$: an image are random processes which are characterised by a joint probability density function (pdf). A discrete 1D random process is said to be **stationary or homogeneous in the strict sense** if the joint probability density function does not depend on the origin of the index n: $pdf(n_a)=pdf(n_b)$. Most of the signals we consider will be **stationary in the wide sense** : In stead of the very general requirements on the probability density function only the mean value and the correlation function will be involved : $E[f(n)] = m_f$ ($= C^{st}$) means that the expectation of f(n) or its mean value is constant . $E[f(k)f^*(k-n)] = R_f(n)$ for all $k \in \mathbf{Z}$ means that the autocorrelation function is only dependent on the delay n.(f^* : complex conjugate of f). The special case of n = 0 reshapes this expression into $R_f(0) = \sigma_f^2$ ($= C^{st}$). This implies that the variance of a stationary signal is also a constant.

Non-stationary signals are described by their probability density function. When the samples are known there is no problem to send a non stationary signal trough a stationary system and normal convolution provides the result. When only the mean value and the variance are known the convolution operation results in some simple expressions (see 2.3.2). Specific techniques for non stationary convolution and deconvolution will be developed in the Chapters 3 and 5.

2.3. Convolution

2.3.1. 1D Convolution

The operation of sending a signal through a system is called a stationary convolution when the system function itself does not change during the operation. The simplest signal to 'interrogate' a system is a delta pulse. To describe the operation , in engineering terms phrased as "sending a signal through a system" one first needs the

mathematical shifting operation : $\delta(t)$ is the delta pulse at the origin (t = 0); to delay the delta pulse with τ time units, the expression should be modified to $\delta(t - \tau)$ which give a delta pulse at $t = \tau$. τ can be positive or negative and should be considered on the interval $-\infty$ to $+\infty$. For causal systems we start from 0. To 'catch' the specific value of a system, one should take $h(\tau).\delta(t - \tau)$. Finally, all these values are 'collected' with the integral :

$$h(t) = \int_{-\infty}^{\infty} h(\tau)\delta(t-\tau)d\tau \qquad (2.1)$$

This integral operation becomes a summation expression with discrete functions :

$$h(n) = \sum_{k=-\infty}^{\infty} h(k)\delta(n-k) \qquad (2.2)$$

Because of this "interrogation" operation h(t) and h(n) are called the **continuous and discrete impulse responses of the system.**

Considering the fact that signals and systems can be constructed as linear combinations of delta pulses , the convolution of a more general function x(t) (or x(n)) with a system h(t) (or h(n)) can de described as

$$y(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau \quad (\text{ or } \quad y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) \ ) \qquad (2.3)$$

The operation is commutative. A shorthand notation makes use of the operator $*$

$$y(t) = h(t) * x(t) = x(t) * h(t) \quad (\text{ or } \ y(n) = h(n) * x(n) = x(n) * h(n) \ ) \qquad (2.4)$$

2.3.2. Convolution of Non Stationary Signals

When only the mean or variance of a signal is known ,the convolution ends in a simple multiplication. Without proof we note that :

$$mean_{output} = \sum_{n=-\infty}^{\infty} h(n).mean_{input} \tag{2.5}$$

$$\sigma^2_{output} = \sum_{n=-\infty}^{\infty} h^2(n).\sigma^2_{input} \tag{2.6}$$

Causal systems start from n = 0.

2.3.3. 2D Convolution

Theorems for 1D convolution can be transposed to 2D. Another terminology is sometimes used ; the input signal will be called the object function, instead of an impulse response one speaks of a point spread function (psf) and the output signal is called the blurred object . [59] describes extensively the 2D convolution and its applications. The operation is defined by the following expressions:

$$y(n_1,n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h(k_1,k_2)x(n_1-k_1,n_2-k_2) \tag{2.7}$$

$$y(n_1,n_2) = h(n_1,n_2) ** x(n_1,n_2) \tag{2.8}$$

The same expressions as (2.5) and (2.6) can de derived for the propagation of mean values and variances through a stationary point spread function. To keep for instance the mean value of an image constant in a convolution operation, the values of $h(n_1,n_2)$ are chosen in such a way that the sum of all $h(n_1,n_2)$ is equal to 1.

2.4. Algebraic Modelling of the Convolution Operation

2.4.1. 1D Operation

The discrete convolution can also be performed with a matrix operation. To do so a circulant matrix H must be constructed with the delayed function. The simplest way to explain this is with an example : Consider a system $h(n) = \alpha^n$, $n \in \{0,1,\ldots N-1\}$ (N=3) and a signal $x(n) = \{1,1,1,0,0,0\}$ then $g(n) = h(n) * x(n) = \{1, 1+\alpha, 1+\alpha+\alpha^2, \alpha+\alpha^2, \alpha^2\}$. The matrix construction for the convolution results in :

$$H = \begin{bmatrix} 1 & 0 & 0 & \alpha^2 & \alpha & 1 \\ \alpha & 1 & 0 & 0 & \alpha^2 & \alpha \\ \alpha^2 & \alpha & 1 & 0 & 0 & \alpha^2 \\ 0 & \alpha^2 & \alpha & 1 & 0 & 0 \\ 0 & 0 & \alpha^2 & \alpha & 1 & 0 \end{bmatrix} \qquad x = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ \alpha + 1 \\ \alpha^2 + \alpha + 1 \\ \alpha^2 + \alpha \\ \alpha^2 \end{bmatrix}$$

$$H \quad . x = \quad y \tag{2.9}$$

$$\begin{bmatrix} 1 & 0 & 0 & \alpha^2 & \alpha & 1 \\ \alpha & 1 & 0 & 0 & \alpha^2 & \alpha \\ \alpha^2 & \alpha & 1 & 0 & 0 & \alpha^2 \\ 0 & \alpha^2 & \alpha & 1 & 0 & 0 \\ 0 & 0 & \alpha^2 & \alpha & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ \alpha + 1 \\ \alpha^2 + \alpha + 1 \\ \alpha^2 + \alpha \\ \alpha^2 \end{bmatrix}$$

Note the circulant construction of the matrix H . In the successive columns a delayed and circulant version of the system elements is reproduced . The number of columns is dependent on the number of elements in the discrete signal (M). The

number of rows is dependent on the number of elements in the convolution result (M+N-1).

When the columns of the H matrix are not composed of the same elements of h(n); for example when α would take another value in every column, the system is called non stationary and the operation is a non stationary convolution. A further discussion of this problem is given in Chapter 3 .

2.4.2. 2D Operation

The digitised object function $x(n_1,n_2)$ and the point spread function $h(n_1,n_2)$ of sizes $A \times B$ and $C \times D$ are extended to $M \times N$ by padding them with zeros. With $M \geq A + C - 1$ and $N \geq B + D - 1$ overlap is avoided . When $M = \max(A,C)$ and $N = \max(B,D)$ we call the operation a circular convolution . This result in an overlap at the edges of the image, if $A << C$ and $B << D$ the effect is negligible.

To construct the H matrix of dimension $MN \times MN$, M^2 partition should be considered with each partition of a size $N \times N$.

$$H = \begin{bmatrix} H_0 & H_{M-1} & H_{M-2} & \cdots & H_1 \\ H_1 & H_0 & H_{M-1} & \cdots & H_2 \\ H_2 & H_1 & H_0 & \cdots & H_3 \\ \vdots & H_2 & H_1 & \cdots & H_4 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ H_{M-1} & H_{M-2} & H_{M-3} & \cdots & H_0 \end{bmatrix}_0 \qquad (2.10)$$

Each partition H_j is constructed from the j^{th} row of the extended point spread function $h(n_1,n_2)$:

$$H_j = \begin{bmatrix} h_{j,0} & h_{j,N-1} & h_{j,N-2} & \cdots & h_{j,1} \\ h_{j,1} & h_{j,0} & h_{j,N-1} & \cdots & h_{j,2} \\ h_{j,2} & h_{j,1} & h_{j,0} & \cdots & h_{j,3} \\ \vdots & h_{j,2} & h_{j,1} & \cdots & h_{j,4} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ h_{j,N-1} & h_{j,N-2} & h_{j,N-3} & \cdots & h_{j,0} \end{bmatrix}_{\square} \qquad (2.11)$$

As with the impulse response for the 1D case, the point spread function for the 2D case will be considered to be periodic. H_j is a circulant matrix, and the blocks of H are subscribed in a circular manner. For these reasons, the matrix H in Eq.(2.9) is called a block-circulant matrix.

Equation (2.9) still holds , but the dimension of becomes very loose. For M=N=256 the size is $M^2 \times N^2 = 262{,}144 \times 262{,}144$.

2.5. Transfer Function

A very interesting signal function to convolve a stationary system with is the complex exponential $e^{j\omega t}$ (or for the discrete case $e^{j\Omega n}$) because this signal function stays invariant under convolution :

$$\int_{-\infty}^{\infty} h(\tau) e^{j\omega(t-\tau)} d\tau = e^{j\omega t} \int_{-\infty}^{\infty} h(\tau) e^{-j\omega\tau} d\tau$$

$$= e^{j\omega t}. H(\omega)$$

For the purpose of official definitions one interchanges τ for t. This is not a problem however, because they both represent time.

$$H(\omega) = \int_{-\infty}^{\infty} h(t)e^{-j\omega t}dt \qquad (2.12)$$

$e^{j\omega t}$ is an eigenfunction for the operation and $H(\omega)$ are the eigenvalues. Note that it is a frequency domain function. A more general name for $H(\omega)$ is the transfer or system function. In the discrete case the convolution operation becomes

$$\sum_{k=-\infty}^{\infty} h(k)e^{j\Omega(n-k)} = e^{j\Omega n}\sum_{k=-\infty}^{\infty} h(k)e^{-j\Omega k}$$

$$= e^{j\Omega n} \cdot H(e^{j\Omega})$$

The same remark applies for the continuous function : Switch k for n. This is important because it sometimes leads to confusion about the variable k which in the Discrete Fourier Transform (DFT) is used as an integer frequency counter while n is used as an integer time counter!

$$H(e^{j\Omega}) = \sum_{n=-\infty}^{\infty} h(n)e^{-j\Omega n} \qquad (2.13)$$

$e^{j\Omega}$ is once again an eigenfunction and $H(e^{j\Omega})$ could be considered as eigenvalues. $H(e^{j\Omega})$ is called the transfer function of the discrete system h(n). $H(e^{j\Omega})$ is continuous and periodic. Two problems need to be solved to come to a matrix notation for (2.13)

(i) The sum has to be made finite and the impulse response assumed periodic :

$$h(n) = h(n + m.N) \text{ with } m \in \mathbf{Z} \quad \text{and} \quad \sum_{-\infty}^{\infty} \rightarrow \sum_{0}^{N-1}$$

(ii) The complex exponential expression has to be made discrete : $\Omega \rightarrow 2\pi.k/N$

This results in the Discrete Fourier Transform (DFT) :

$$H(k) = \sum_{n=0}^{N-1} h(n)e^{-j2\pi kn/N} \qquad (2.14)$$

This is a very intuitive approach and it helps developing practical solutions!

2.6. Fourier Transforms

The definition of the transfer function is in fact identical to the one used for the Fourier Transform. In this section we will restrict ourselves to review the transforms for 4 different cases of stationary signals Non stationary signal transforms are treated in Chapter 5. The theory can be found in [63] . Here we focus on a few very special properties of the transforms. We will use a square wave and show the differences and similarities for all of the following four cases :

(i) continuous non-periodic signals/systems are transformed with the Fourier Integral.

(ii) continuous periodic signals/systems are analysed with the Fourier Series.

(iii) discrete non-periodic signals/systems are analysed with the Discrete Fourier Series.

(iv) discrete periodic signals/systems are transformed with the Discrete Fourier Transform.

2.6.1. Fourier Integral

We consider a unique pulse $h(t) = 1$ for $0 \le t \le t_0$

$= 0$ for $t_0 < t \le 1$.

In our example we take $t_0 = .5$.

$\omega := 0..32$ $\quad$ $h(t) := if(t \le .5, 1, 0)$

$$H(\omega) := \int_0^{\infty} h(t) \cdot e^{j \cdot \omega \cdot t} dt \qquad (2.15)$$

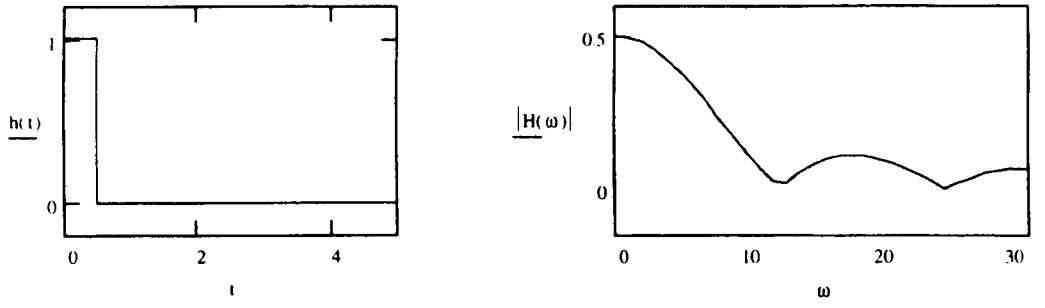


Fig.2.2. Non periodic analogue pulse and its non periodic analogue amplitude spectrum $t_0.sinc(\omega.t_0/2)$ is the result of the Fourier integral for the amplitude response.

Remark : The Fourier transform of a non-periodic continuous function in time domain results in a non-periodic continuous function in frequency domain. The inverse transform can also be performed with an integral and is therefore numerically not of interest.

2.6.2. Fourier Series

A periodic version of the same pulse can be decomposed into the following series. In this example, we consider a periodic pulse with a width $t_0 = .5$ and a period $T = 4$

$n := 0..20$ $\quad T := 4$ $\quad h(t) := if(t \le .5, 1, 0)$

$$C_n := \frac{1}{T} \cdot \int_0^T h(t) \cdot e^{\frac{-j \cdot 2 \cdot \pi \cdot n \cdot t}{T}} dt \qquad (2.16)$$

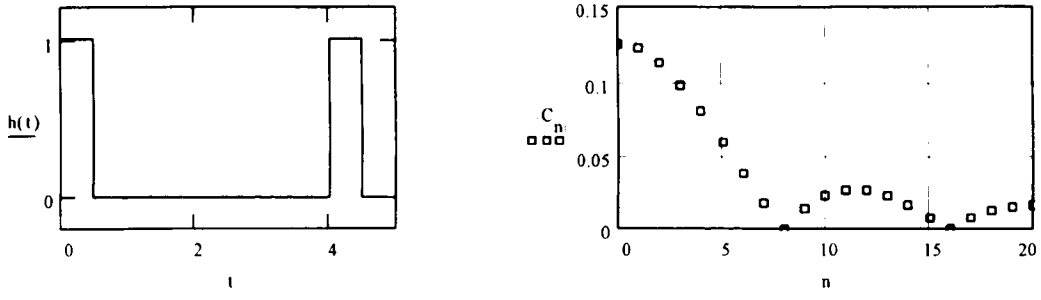


Fig.2.3Due to the periodization of the signal the spectrum has become discrete. The amplitude of the discrete spectrum is described by : (t0/T).sinc(π.n.t0/T).

Remark : Composing the original signal back with the harmonics is possible with an infinite sum formula. Numerically, this is not possible without errors and thus far from practical!

2.6.3. Discrete Fourier Series

A discrete version of the pulse can be made by sampling the analogue pulse at sample intervals t_s. In our example we choose t_s = .1 sec. This brings the number of non-zero samples in the discrete series to M =6.

$$n \quad 0..100 \qquad \Omega \quad 0, \frac{\pi}{20} .. 2\cdot\pi \qquad h(n) \quad if(n < 5, 1, 0)$$

$$H(\Omega) \quad \sum_{n=0}^{\infty} h(n)\cdot e^{j\cdot n\cdot\Omega}$$

(2.17)

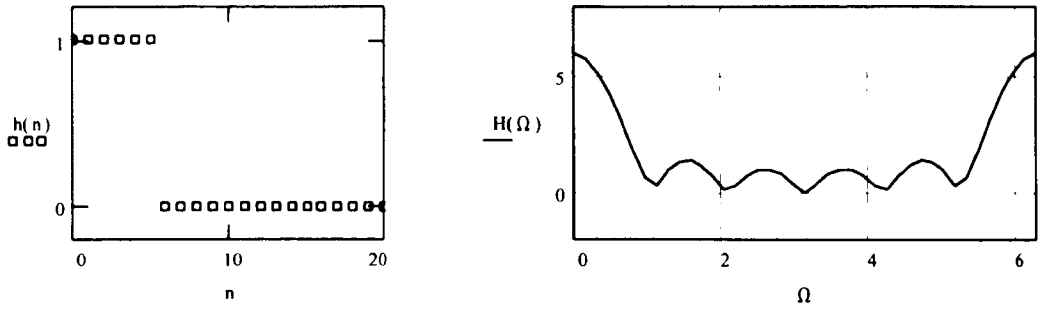


Fig. 2.4. The spectrum of a discrete pulse is continuous. The amplitude response is characterised by the expression : sin(Ω M/2)/sin(Ω/2)

Remark : Periodicity has come into the spectrum.

2.6.4. Discrete Fourier Transform (DFT)

Fourier and discrete Fourier series have to be combined to produce a discrete finite signal in both spaces. The fact that one has to accept that it is periodic in both spaces can be annoying and special contra measures should be taken when using DFTs for analysing non-periodic signals. In this example a pulse is analysed . As one has to accept that it must be considered as periodic, errors compared with the Fourier integral results will gradually increase as one reaches half sampling frequency (k=20).

$N := 40 \qquad n := 0..N-1 \qquad k := 0..N-1$

$h(n) := if(n<5,1,0)$

$$H(k) := \frac{1}{N}\cdot\sum_{n=0}^{N-1} h(n)\cdot e^{\frac{j\cdot 2\pi\cdot n\cdot k}{N}}$$

(2.18)

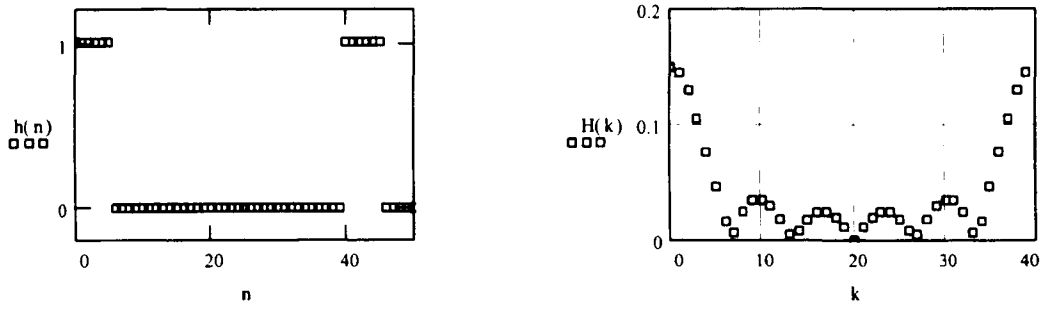


Fig.2.5. The amplitude response in the frequency domain is characterised by : $\sin(\pi.M.k/N) / N.\sin(\pi.k/N)$

Remark : Redundancy can be removed from the calculation of the DFT . By considering powers of two as the number of samples involved in the DFT the complexity of calculation can be reduced from N^2 operations to $(N/2)\log_2 N$. The DFT then becomes FFT (Fast Fourier Transform).

2.6.5. 2D DFT

All 1D transforms can be transposed to equivalent 2D transforms. The DFT is the only usable transform for numerical applications. Therefore we only consider this transform and refer for a theoretical background to [59]. The 2 DFT can be defined as :

$N1 := 16 \quad N2 := 16$

$n1 := 0..N1-1 \quad n2 := 0..N1-1$

$k1 := 0..N1-1 \quad k2 := 0..N1-1$

$$x_{n1,n2} := \text{if}\left(\frac{N1}{4} \le n1 < N1 \cdot 5, \text{if}\left(\frac{N1}{4} \le n2 < N1 \cdot 5, 1, 0\right), 0\right)$$

$$X_{k1,k2} := \sum_{k1=0}^{N1-1} \sum_{k2=0}^{N2-1} x_{n1,n2} \cdot e^{\frac{j \cdot 2 \cdot \pi \cdot k1 \cdot n1}{N1}} \cdot e^{\frac{j \cdot \pi \cdot k2 \cdot n2}{N2}}$$

(2.19)

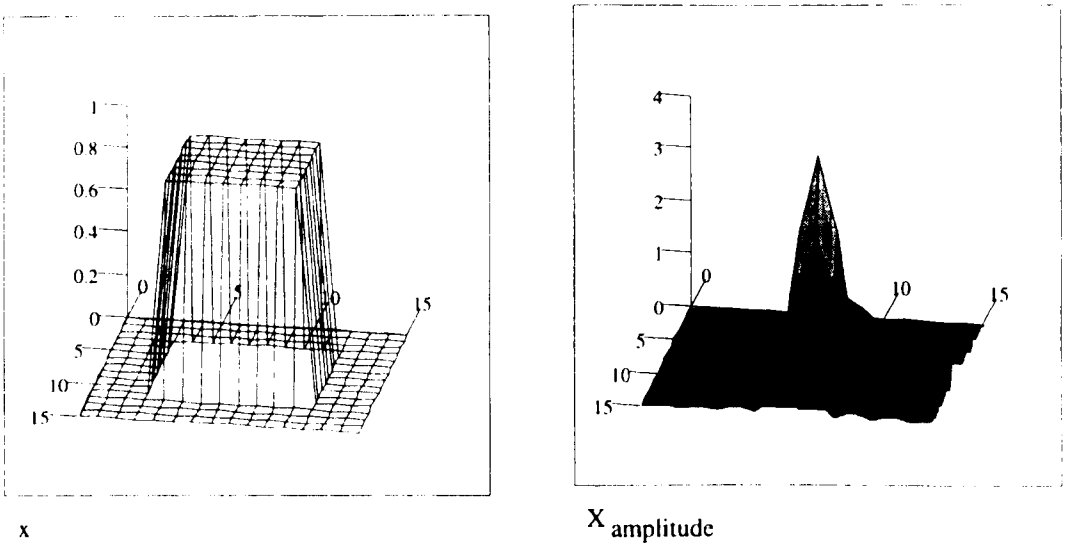


Fig.2.6. Spatial pulse and its 2D DFT amplitude representation . A shift operation has been performed on the spectral data so that the zero frequencies are translated to the middle of the k_1,k_2 plane.

2.6.6. Discrete Cosine Transform (DCT)

The Discrete Cosine Transform is extensively used in image coding, especially in JPEG. It can be used to obtain a more compact data information on images before sending it through a neural network for identification. Once again we refer to [59] for an in depth exposé on the subject. The transform is defined as :

$$C_x(k) = \sum_{0}^{N-1} 2x(n)\cos(\pi N / 2)k(2n+1) \qquad \text{for } 0 \le k \le N-1 \qquad (2.20)$$

$$= 0 \text{ otherwise}$$

One of the major advantages is the fact that if x(n) is real, $C_x(k)$ is also real. Thus it allows one to work with real instead of complex frequency data. It not only halves memory space but due to the very compact form of the transformed data it can for

instance successfully be used as an input for neural networks to identify objects with much less data. (See Chapter 8)

$n := 0..31$

$k := 0..31$ $\quad x_n := if(n \le 7, 1, 0)$ $\quad C_x := costr(x)$

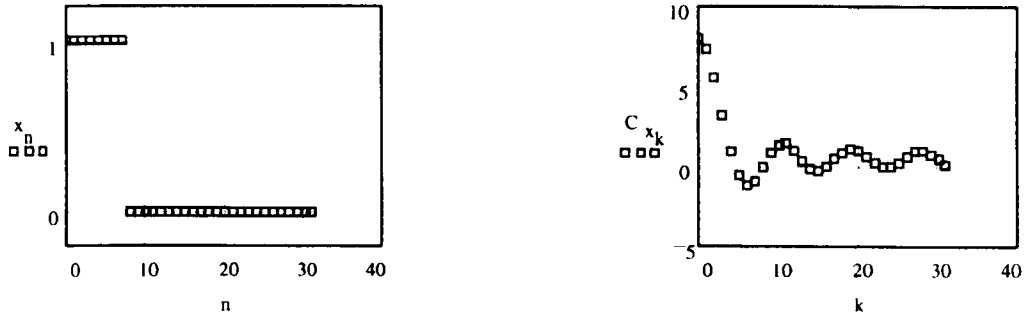


Fig.2.7. Cosine transform of square wave pulse.

The 2D Discrete Cosine Transform is defined by :

$$C_x(k_1,k_2) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} 4x(n_1,n_2)\cos((\pi N_1/2)k_1(2n_1+1))\cos((\pi N_2/2)k_2(2n_2+1))$$

$$\text{for } 0 \le k_1 \le N_1 \ , 0 \le k_2 \le N_2$$

$$= 0 \quad \text{otherwise} \qquad (2.21)$$

2.6.7. Duality

Although the inverse Fourier transforms were not defined explicitly in this chapter, one can find out [59] that there is definite symmetry in the equations. They are similar but not quite identical in form. This phenomena is called the duality in the Fourier transform (FT). An illustration is found in Fig.2.8 . It shows that :

square wave in time domain → FT → sinc function in frequency domain

sinc function in time domain → FT → square wave in frequency domain

Note that the square wave (Fig. 2.8a.) is dimensionally the same as in Fig. (2.8d.). The duty cycle of the square wave in both domains is the same

($t_0/T=.5/4=.125 \quad = \Omega_0/\Omega_s=2 \times .392/2\pi=.1248$). This principle proves very useful in the design of FIR (Finite Impulse Response) filters.

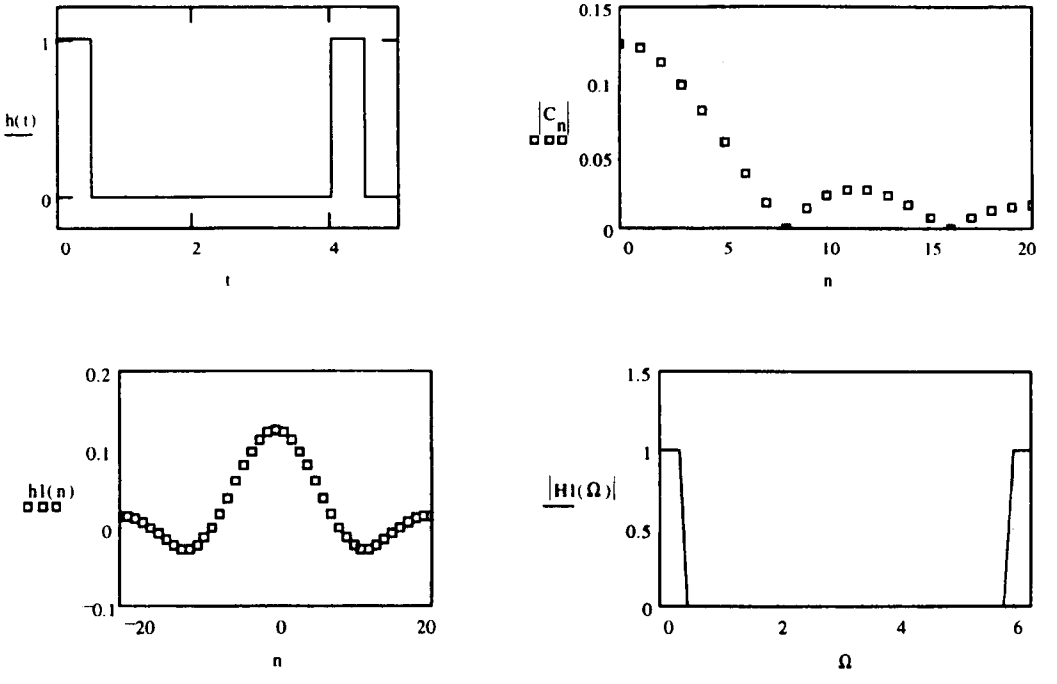


Fig.2.8. Illustration of the duality principle of Fourier transforms.

2.7. Diagonalization of the Circulant Matrix

2.7.1. 1D Operation

We now consider the matrix multiplication of the circulant matrix H and a discrete function w(k).

$$w(k) = \begin{bmatrix} 1 \\ e^{\frac{j2\pi k}{N}} \\ e^{\frac{j2\pi 2k}{N}} \\ \vdots \\ \vdots \\ \vdots \\ e^{\frac{j2\pi k(N-1)}{N}} \end{bmatrix}$$

It can be shown that :

$$H.w(k) = \lambda(k).w(k) \qquad (2.22)$$

This expression indicates that w(k) is an eigenvector of the circulant matrix H and that $\lambda(k)$ is its corresponding eigenvalue. Next , let us form an N × N matrix W by using the N eigenvectors of H as columns :

$$W = [w(0)\ w(1)\ w(2)\ \ldots\ w(N-1)] \qquad (2.23)$$

The kn^{th} element of W , denoted by W(k,n) is given by

$$W(k,n) = e^{j2\pi kn/N} \qquad \text{for } k,n = 0,1,2, \ldots , N-1.$$

The orthogonality properties of the complex exponential allows writing the inverse matrix W^{-1} , by inspection ; its kn^{th} element, symbolised as $W^{-1}(k,n)$ is

$$W^{-1}(k,n) = (1/N) . e^{-j2\pi kn/N}$$

Multiplying these 2 equations :

$$WW^{-1} = W^{-1}W = I \qquad (2.24)$$

where I is the M × M identity matrix. The importance of the existence of the inverse matrix W^{-1} is that it guarantees that the columns of W (the eigenvectors of H) are linearly independent. Elementary matrix theory tells us that H may then be expressed in the form

$$H = W D W^{-1} \tag{2.25}$$

or $$D = W^{-1} H W \tag{2.26}$$

where D is a diagonal matrix whose elements D(k,k) are the eigenvalues of H ; that is

$$D(k,k) = \lambda(k) \tag{2.27}$$

Equation (2.25) indicates that H is diagonalized by using W^{-1} and W as expressed in eq. (2.26).

2.7.2. 2D Operation

The diagonalizing process of a block-circulant matrix is similar to the 1D case. This is however not the subject of further research.

2.8. Convolution property

Signals convolve in time with systems and result in a new signals. It is interesting to find out how their respective transfer functions act upon each other :

$$y(t) = h(t) * x(t) \qquad \text{or discrete : } h(n) = h(n) * x(n) \tag{2.28}$$

Taking the system function of both sides of the equality results in

$$Y(\omega) = \int_{t=-\infty}^{\infty} \int_{\tau=-\infty}^{\infty} h(\tau)x(t-\tau)d\tau e^{-j\omega t} dt$$

After some mathematical manipulations, which detailed explanation can be found in proofs of the convolution property [63] , the expression becomes :

$$Y(\omega) = H(\omega) \,.\, X(\omega) \qquad (\text{ or discrete } Y(e^{j\Omega}) = H(e^{j\Omega}) \,.\, X(e^{j\Omega}) \;) \tag{2.29}$$

The DFT produces : $$Y(k) = H(k) \,.\, X(k) \tag{2.30}$$

In plain words, this means that a convolution operation in the time domain is transformed into a multiplication in frequency domain.

These expressions can easily be transposed to 2D. For the 2D DFT we have

$$Y(k_1,k_2) = H(k_1,k_2) \,.\, X(k_1,k_2) \qquad (2.31)$$

2.9. Modulation Property

Similar to the convolution property (see also (63) one can proof that for all Fourier transforms the multiplication of two signals in the time domain results in the convolution of the 2 transformed signals in frequency domain :

$h_1(t) \,.\, h_2(t) \rightarrow H_1(\omega) * H_2(\omega)$ (or discrete $h_1(n) \,.\, h_2(n) \rightarrow H_1(e^{j\Omega}) * H_2(e^{j\Omega})$)

(2.32)

2.10. Deconvolution

Eq. (2.29) can also be rewritten as :

$$X(\omega) = Y(\omega)/H(\omega) \qquad (\text{ or discrete } X(e^{j\Omega}) = Y(e^{j\Omega})/H(e^{j\Omega}) \;) \qquad (2.33)$$

The DFT case enables a numerical solution : $X(k) = Y(k)/H(k)$ (2.34)

In 2D we have $X(k_1,k_2) = Y(k_1,k_2)/H(k_1,k_2)$ (2.35)

This means that there is an elegant way to deconvolve an output signal y(t) (or y(n)) from a system h(t) (or h(n)) : Calculate the transfer functions of output and system and divide one by the other. The same argument can be made for the 2D case. However, in practice one is required to regularise the result.

With matrix notation the 1D and 2D deconvolution operation becomes :

$$x = H^{-1} \,.\, y \qquad (2.36)$$

Calculating the inverse matrix is not always possible and can, for large matrices become very computer intensive.

Substituting Eq. (2.25) in (2.36),

$$x = (WDW^{-1})^{-1}y$$

$$= WD^{-1}W^{-1}y \qquad (2.37)$$

Multiplying both sides of the equality with W^{-1} yields :

$$W^{-1}x = D^{-1}(W^{-1}y) \qquad (2.38)$$

This expression is very similar to Eq. (2.33). $W^{-1}x$, $W^{-1}y$ are the respective DFTs of $x(n)$, $y(n)$ for 1D and $x(n_1,n_2)$, $y(n_1,n_2)$ for 2D. D^{-1} is the inverse matrix of the diagonal matrix D - a formidable reduction provided D is indeed a diagonal matrix. This is the case for circulant matrices H constructed from stationary signals. The more non-stationary the system is, the more non-zero elements will show up beside the diagonal and the more it will look like an ordinary matrix with no special properties to produce a fast inversion.

2.11. Laplace Transform

In engineering literature one very often states that $\int_{-\infty}^{\infty} h(t)e^{-j\omega t}dt$ should be $< \infty$,excluding some functions for the Fourier transform . The Fourier transform of the unit step function for example, results in $\pi\delta(\omega) + 1/j\omega$. Problem is $\delta(\omega)$ where $\delta(\omega)\rightarrow\infty$ when $\omega\rightarrow 0$. This can be omitted by generalising the Fourier transform to complex variables or practically, introducing a decaying function ($e^{-\sigma t}$) that has to be multiply with h(t) so that $\int_{-\infty}^{\infty} h(t)e^{-\sigma t}e^{-j\omega t}dt$ shows no more delta function in its solution. Defining $s = \sigma+j\omega$ results in a new transform variable and transform

$$H(s) = \int_{-\infty}^{\infty} h(t)e^{-st}dt \qquad (2.39)$$

This is the Laplace Transform. All properties derived for the Fourier transform are also valid for the Laplace transform. To the series of expressions (2.15), (2.16) we add

$$Y(s) = H(s).X(s) \qquad (2.40)$$

and

$$H(s) = Y(s)/X(s) \qquad (2.41)$$

Note that the variable s is complex . This makes H(s) a rather 'complex' function. Values of s where H(s) is zero or infinite are very interesting. They are respectively called the zeros and poles in the s-plane of the system function. Stable systems will have poles in the left part ($\sigma<0$) of the s-plane. To illustrate this we consider a transfer function with 2 poles (c and d)and 1 zero (a).

$\sigma := -20, -1..0$ $\omega := -25, -1..25$

$a := 10.45$

$c := 15.45$

$d := 5.45$

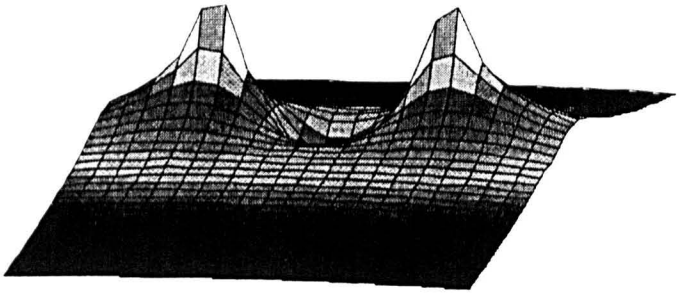
$s(\sigma,\omega) := \sigma + j\cdot\omega$

$$H1(\sigma,\omega) := \frac{s(\sigma,\omega)+a}{s(\sigma,\omega)+c}\cdot\frac{1}{s(\sigma,\omega)+d}$$

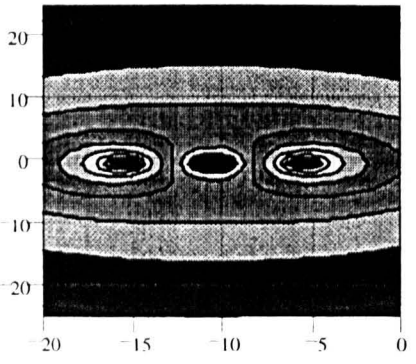
$i := 0..20$ $j := 0..51$

$M1_{j,i} := \log(|H1(i-20, j-25)|)$

$M2_{i,j} := M1_{j,i}$



M1



M2

Fig.2.10. M1 shows the spatial representation of a transfer function in s with 2 poles at s_1=-5.45 and s_2=-15.45 and 1 zero at s_3=-10.45. The contour plot of M2 makes this clear.

Laplace transforms are used extensively in filter design and control engineering.

2.12. Z Transform

2.12.1 1D Z Transform

Similar to the Fourier transform for analogue signals we generalise the discrete Fourier transform of signals or systems. We can introduce a discrete decaying function ρ^{-n} . This reshapes expression (2.13) into $\sum_{n=-\infty}^{\infty} h(n)\rho^{-n}e^{-j\Omega n}$. With a new variable z = $\rho\, e^{j\Omega}$ we obtain a new transform called the z transform :

$$H(z) = \sum_{n=-\infty}^{\infty} h(n)z^{-n} \qquad (2.42)$$

The convolution property is also valid for the z transform ,.i.e.

$$Y(z) = H(z) \,.\, X(z) \qquad (2.43)$$

and

$$H(z) = Y(z)/X(z) \qquad (2.44)$$

As with the Laplace transform, we have a complex variable in a complex plane. The position of poles and zeros of H(z) are important to examine the stability of systems. $\rho e^{j\Omega}$ describes the z plane in polar co-ordinates. Discrete systems are said to be stable when their poles lie inside the unit circle defined by ρ=1. A similar construction to the Laplace transform (Fig. 2.10) can be made.

Z-transforms are of importance in describing and designing of digital filters and digital control systems.

2.12.2. 2D Z Transform

When the spectrum of a point spread function $h(n_1,n_2)$ has to be calculated, the 2D DFT does not always give a finite result. As with the 1D z transform we have to consider the finiteness of $\sum_{n_1=-\infty}^{\infty}\sum_{n_2=-\infty}^{\infty} h(n_1,n_2)$ and introduce a damping factor $\rho_1\rho_2$ to make the transform possible. Eq (2.6) translated to 2D becomes

$$\sum_{n_1=-\infty}^{\infty}\sum_{n_2=-\infty}^{\infty} h(n_1,n_2)\rho_1\rho_2 e^{-j\Omega_1 n_1} e^{-j\Omega_2 n_2}$$

With $z_1 = \rho_1 e^{-j\Omega 1.n1}$ and $z_2 = \rho_2 e^{-j\Omega 2.n2}$ we construct the 2D z transform :

$$H(z) = \sum_{n_1=-\infty}^{\infty}\sum_{n_2=-\infty}^{\infty} h(n_1,n_2) z_1^{-n_1} z_2^{-n_2} \tag{2.45}$$

These functions are 4 dimensional and rather difficult to depict. Stability considerations are also much more difficult than in the 1D case. Although one could also speak of pole planes and zero planes we will only consider systems with zero planes because they are much more stable and can be implemented with a 2D convolution operation.

2.13. The Chirp Z Transform

The FFT imposes severe restrictions on the number of samples and the analysis contour . An efficient algorithm to analyse a finite duration sequence by evaluating its z transform along certain general contours in the z plane is the chirp z-transform.[64] It is about 3 times slower than the FFT for the same amount of data, but it has some major advantages. The number of samples in the original sequence and in the transformed sequence can be different and it offers the possibility to scan a well defined zone in the z-plane.

Let x(n) be a given N-point sequence with z-transform

$$X(z) = \sum_{n=0}^{N-1} x(n) z^{-n} \tag{2.46}$$

The z-transform becomes a DFT by considering the transform on the unit circle and taking $z_k = e^{j2\pi k/N}$ with k =0..N-1.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N} \tag{2.47}$$

With the Chirp Z Transform (CZT) the z-transform can be evaluated along a more general contour : $z_k = A.W^{-k}$ with $k = 0...M-1$ (2.48)

M being an arbitrary integer , not necessarily equal to N, and A and W are complex numbers defined with their polar co-ordinates :

$$A = A_0.e^{j2\pi\theta_0} \quad \text{and} \quad W = W_0.e^{j2\pi\varphi_0} \tag{2.49}$$

In Fig.2.12 a region, defined by the following parameters, is examined :

$A_0 := 1 \quad W_0 := 1.025 \quad \theta_0 := \frac{-1}{12} \quad \varphi_0 := \frac{1}{36} \quad M := 100 \quad k := 0..M-1$

$z_k := A_0 \cdot (W_0)^{-k} \cdot \exp\left[j \cdot 2 \cdot \pi \cdot (\theta_0 - k \cdot \varphi_0)\right]$

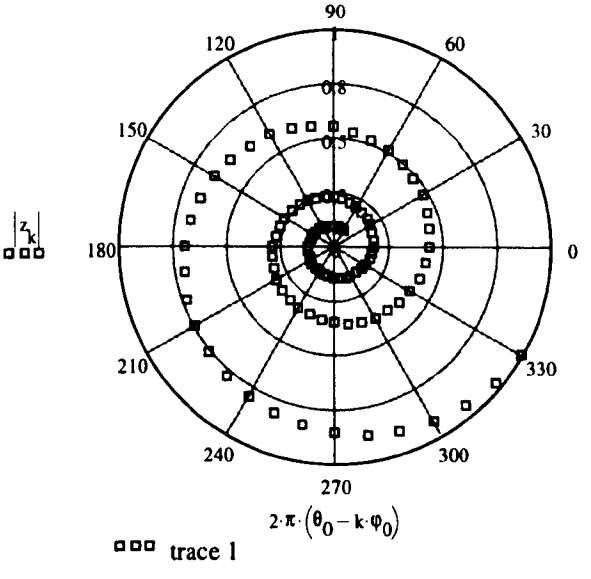


Fig.2.11.

Chirp z-transform can be used to scan for poles and zeros inside the unit circle in the z plane. In this example we start at 330^0 take 100 steps of -10^0 while decreasing the radius from 1 to 0.085.

Note that by an appropriate control of A , W and M all points inside the unit circle could, whitin reasonable limits, be reached. This makes an accurate investigation possible on the values of all poles and zeros in the z transform of a sequence. This is an interesting idea because identifying impulse responses which have poles who are situated near the centre of the unit circle are very difficult to detect because they fade out very quickly.

This can very easily be demonstrated with 2 decaying sine waves x1 and x2. (Fig.2.12.) In x2 it is hardly possible to detect the oscillatory nature of the signal. Their respective FFTs are in fact searching for tracks of poles on the unit circle. The more they are inside this circle the less identifiable they are. The principle of the chirp z-transform can be simplified to an FFT operation with the W_0 parameter integrated in the FFT analysis. It is like scanning the z plane, not on the unit circle but on **circles with radius W_0 inside the unit circle** ! Note how well these transformed signals ($X1_{chirp}$, $X2_{chirp}$) identify the damped oscillations in x1 and x2.

$n := 0..128$ $\alpha 1 := .9$ $\alpha 2 := .6$

$\beta 1 := \frac{\pi}{4}$ $\beta 2 := \frac{\pi}{8}$ $x1_n := \alpha 1^n \cdot \sin(n \cdot \beta 1)$

$x2_n := \alpha 2^n \cdot \sin(n \cdot \beta 2)$

$W0_1 := .9$ $W0_2 := .6$ $x1chirp_n := W0_1^{-n} \cdot \alpha 1^n \cdot \sin(n \cdot \beta 1)$

$x2chirp_n := W0_2^{-n} \cdot \alpha 2^n \cdot \sin(n \cdot \beta 2)$

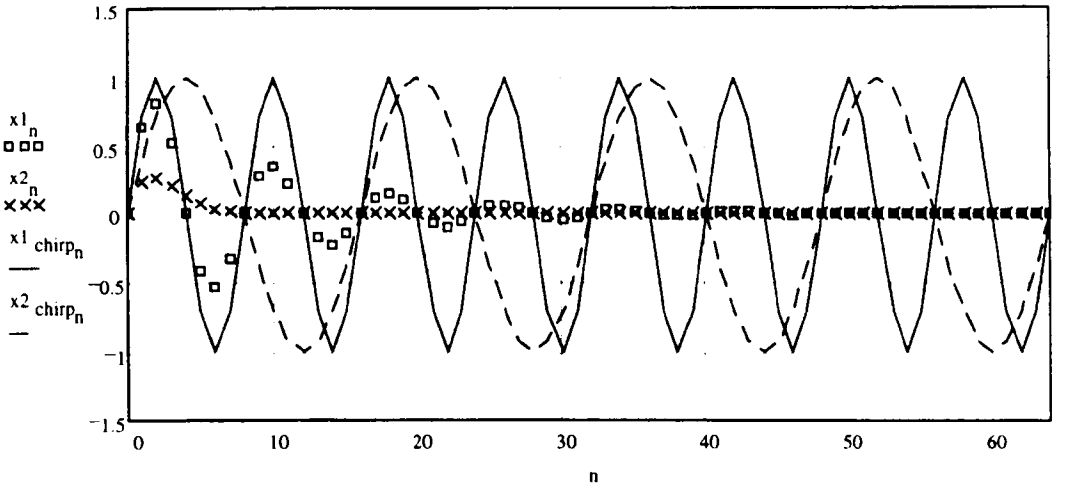


Fig.2.12. Damped oscillations and the result of chirp z techniques (= multiplication with the W_0 parameter) to enhance the signal.

The FFT of the processed signals shows sharply the oscillatory frequencies. It is as if the poles of the signals were lying on the unit circle .

$X1 := cfft(x1)$ $X2 := cfft(x2)$ $X1_{chirp} := cfft(x1_{chirp})$ $X2_{chirp} := cfft(x2_{chirp})$

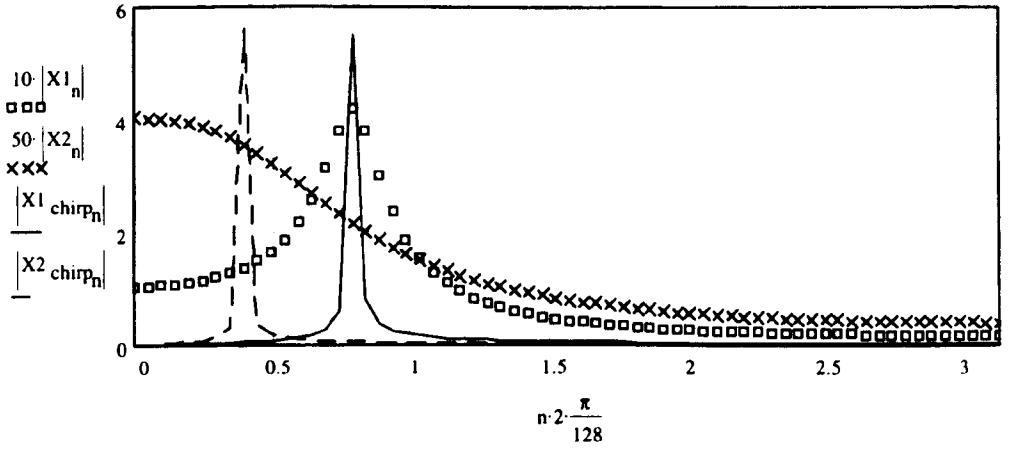


Fig.2.13. FFT's of processed signals shows sharp peaking at π/4 and π/8, in contrast with the inaccurate indication in the classical FFT results.

2.14. Non Stationary Application

When my research started, non stationary deconvolution was the first item I was confronted with. Initially, it seemed to me, that non stationary convolution was in fact a blurring action and that deconvolving had to do with removing the blurring. To translate these problems to usable mathematics the problem was reformulated as : which way was the identifiability of the poles and zeros of a non stationary system affected by blurring ?

This research is not completed because the usefulness of the concept became more and more irrelevant in the evolution of the work. However, I found it interesting enough to report about it :

The usefulness of the chirp z transform will be examined by trying to identify an impulse response x constructed out of 3 individual impulse responses h1, h2 h3 and blurred by a non stationary convolution with a changing sinc function. (In the frequency domain , a varying low pass filter.) When signals are so well identifiable with a chirp z-transform techniques, one can consider in which way they are still 'recognisable' after a masking non-stationary convolution operation.

$$x(n) \rightarrow h(n,n') \rightarrow y(n) \qquad (2.50)$$

Can y(n) be non stationary deconvolved from the non stationary kernel h(n,n') ? We examine this with an example of a 3 pole signal x(n) composed of $h_1(n)$, $h_2(n)$ and $h_3(n)$ and a non stationary kernel h(n,n') (h_n_s : h_non stationary). h_n_s is constructed out of a varying sinc function (see fig.2.14).

$N := 32 \qquad N1 := N - 1 \qquad n := 0..N1 - 1$

$h1_n := .9^n \cdot \sin(n \cdot .5) \qquad h2_n := .8^n \cdot \sin(n \cdot 1) \qquad h3_n := .7^n \cdot \sin(n \cdot 2)$

$x := \text{convol}(h1, \text{convol}(h2, h3)) \qquad N2 := \text{length}(x) \qquad n2 := 0..N2 - 1$

$X := \text{cfft}(x)$

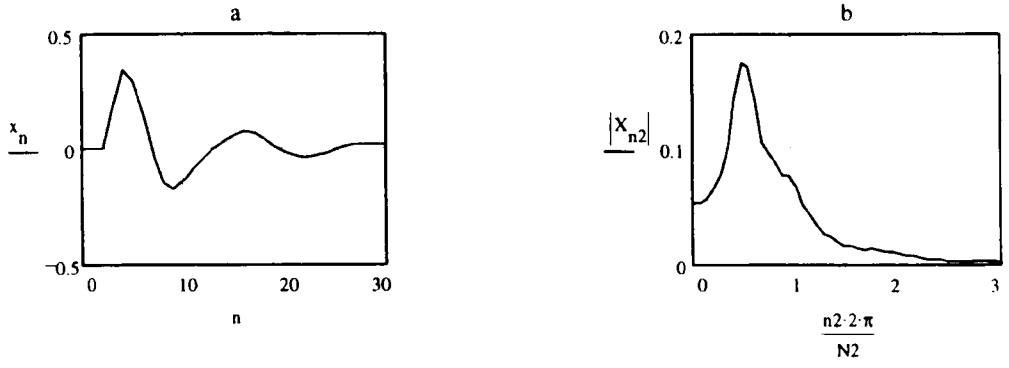


Fig.2.14. Time response (a) and frequency response (b) of a 3 pole system composed of h_1, h_2, h_3.

A non stationary sinc signal is generated with 90 different values. The extreme functions and their respective spectra are shown in Fig. 2.15 and 2.16.

$$\text{sinc}_{lp}(bw,n) := \begin{cases} bw & \text{if } n = 0 \\ bw \cdot \dfrac{\sin(n \cdot \pi \cdot bw)}{n \cdot \pi \cdot bw} & \text{otherwise} \end{cases}$$

$M := 90$

$m := 0..20$

$k := 0..M \qquad \delta := .5 \qquad h\_n\_s_{m,k} := \text{sinc}_{lp}\left[.5 \cdot \left(1 + \delta \cdot \cos\left(k \cdot \frac{\pi}{M+1} \right) \right), m - 10 \right]$

$h\_n\_s_{0_m} := h\_n\_s_{m,0} \qquad h\_n\_s_{.5M_m} := h\_n\_s_{m,\frac{M}{2}} \qquad h\_n\_s_{M_m} := h\_n\_s_{m,M}$

$F\_H_0 := \text{cfft}(h\_n\_s_0) \qquad F\_H_{.5M} := \text{cfft}(h\_n\_s_{.5M}) \qquad F\_H_M := \text{cfft}(h\_n\_s_M)$

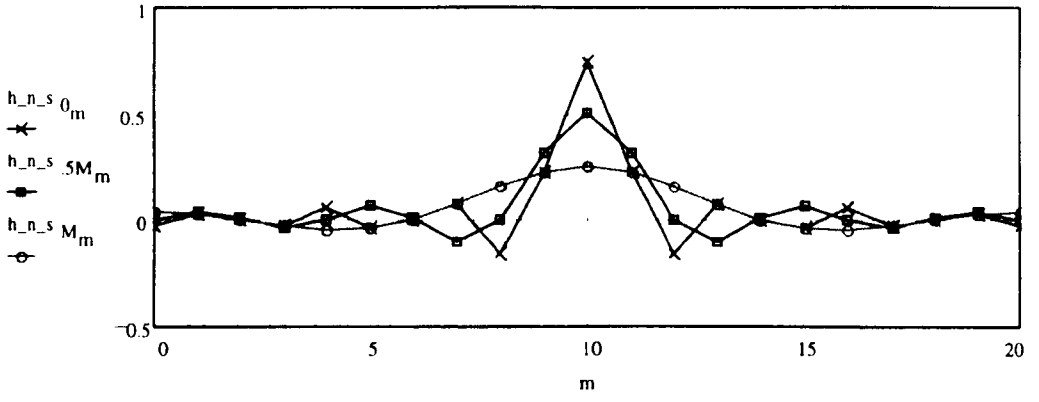


Fig.2.15. Extreme values of the sinc functions.

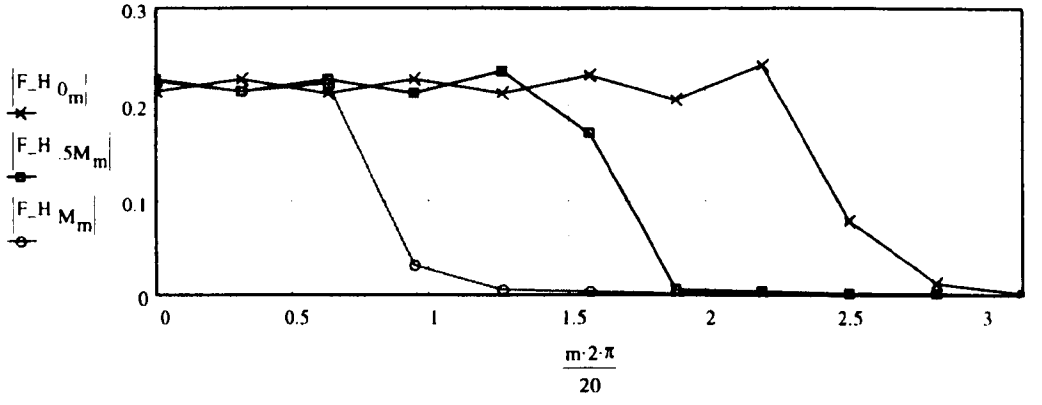


Fig.2.16. Extreme values within which the spectra of the blurring function changes.

To analyse the signal x a matrix H_n_s must be constructed with the non stationary values of the sinc function to perform the non stationary convolution :

y = H_n_s . x .

$z1 := 0..M-1$

$z2 := 0..M$ $\quad zeros_{z1,z2} := 0$

$stac\_z\_h := stack(h\_n\_s, zeros)$

$$modulo(m,k,N) := \left| \begin{array}{l} mod\_count \leftarrow (k+1)\cdot N - k + m \\ mod(mod\_count, N) \quad otherwise \end{array} \right.$$

$ns1 := 0..110$

$ns2 := 0..90$

$H\_n\_s_{ns1,ns2} := stac\_z\_h_{modulo(ns1,ns2,111),ns2}$

$y := H\_n\_s \cdot x$ $\quad Ly := length(y)$ $\quad ly := 0..Ly-1$

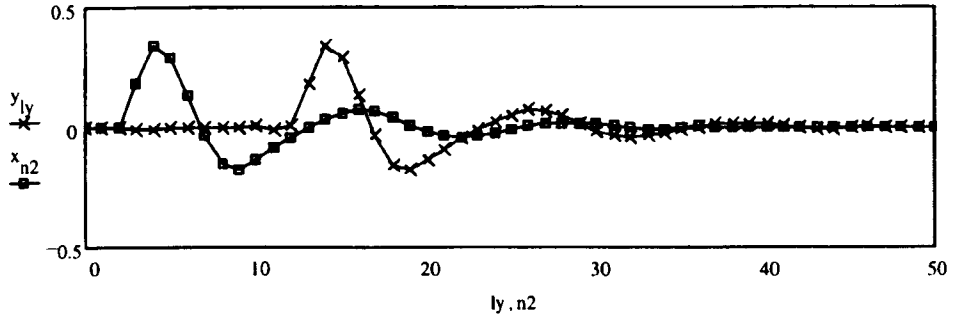


Fig.2.17. The original signal x (conv(h_1,h_2,h_3)) and its blurred version y.

To extract the original contents of the signal, the blurred signal is analysed inside the unit circle. Starting from $\rho = 1$, taking steps of .05 we come to .5 and at all these values, the data is analysed with chirp z techniques. The final result is made by adding all these results together. In comparison with the results in Fig.2.14b one can clearly identify the different frequencies involved in the signal x. The blurring does not harm very much the analysis although the frequency $\Omega = 2$ is hidden by the blurring.

```
chirp_z(h,ρ) := | N ← length(h)
                | for n2 ∈ 0..N - 1
                |   h_{ρ_n2} ← ρ^{-n2} · h_n2
                | cfft(h_ρ)

m_chirpz(h,stp,min_ρ) := | H_mean ← 0
                         | for ρ ∈ 1, 1 - stp .. min_ρ
                         |   | H ← chirp_z(h,ρ)
                         |   | H_mean ← H_mean + H / |H_0|

X_mean := m_chirpz(x, .05, .5)

Y_mean := m_chirpz(y, .05, .5)
```

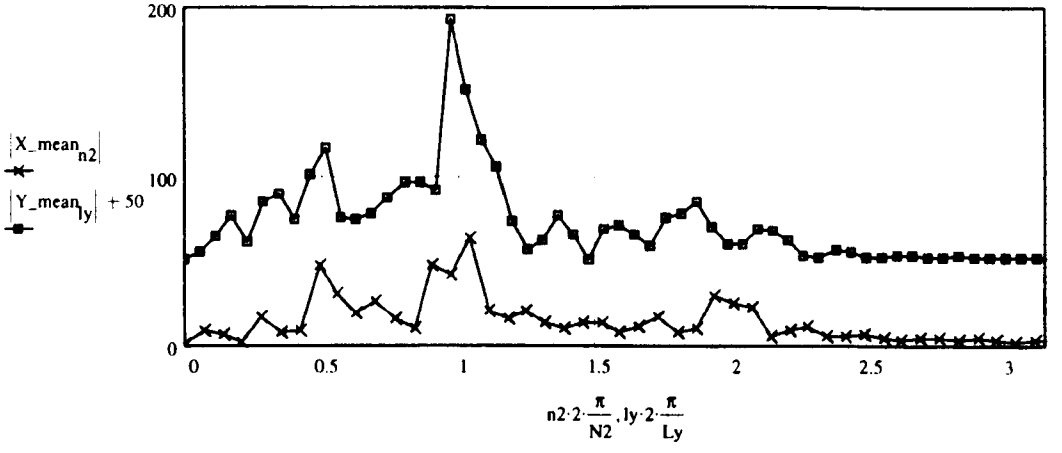


Fig.2.18. Chirp analysis of original and blurred signal

A more difficult problem in the analysis is the identification of the decaying values in the signal. The different chirp analyses have been collected and displayed in Fig.2.19. From the bottom to the top of the picture one sees the analyses from $\rho = 1$ to $\rho = .5$ in steps of 0.1. One can see that when one is close to oscillating frequencies they are very well marked. Further away (at $\rho = 1$ and $\rho = .5$) one sees only weak information about the involved frequencies. Every maximum per function identifies the involved frequency. With $\rho = .9$ there is a maximum on $HH_{ly,2}$ at $\Omega = .5$. The next maximum is on $HH_{ly,4}$ with $\rho = .8$ an $\Omega = 1$. The last maximum is not so well identifiable.

$$\text{max\_chirpz}(h, stp, min\_\rho) := \text{for } \rho \in 1, 1 - stp \,..\, min\_\rho$$
$$\left| \begin{array}{l} H \leftarrow \text{chirp\_z}(h, \rho) \\ H\_norm \leftarrow \dfrac{H}{|H_0|} \text{ if } \rho = 1 \\ H\_norm \leftarrow \text{augment}\left(H\_norm, \dfrac{H}{|H_0|}\right) \text{ otherwise} \end{array} \right.$$

$HH := \text{max\_chirpz}(y, .05, .5)$

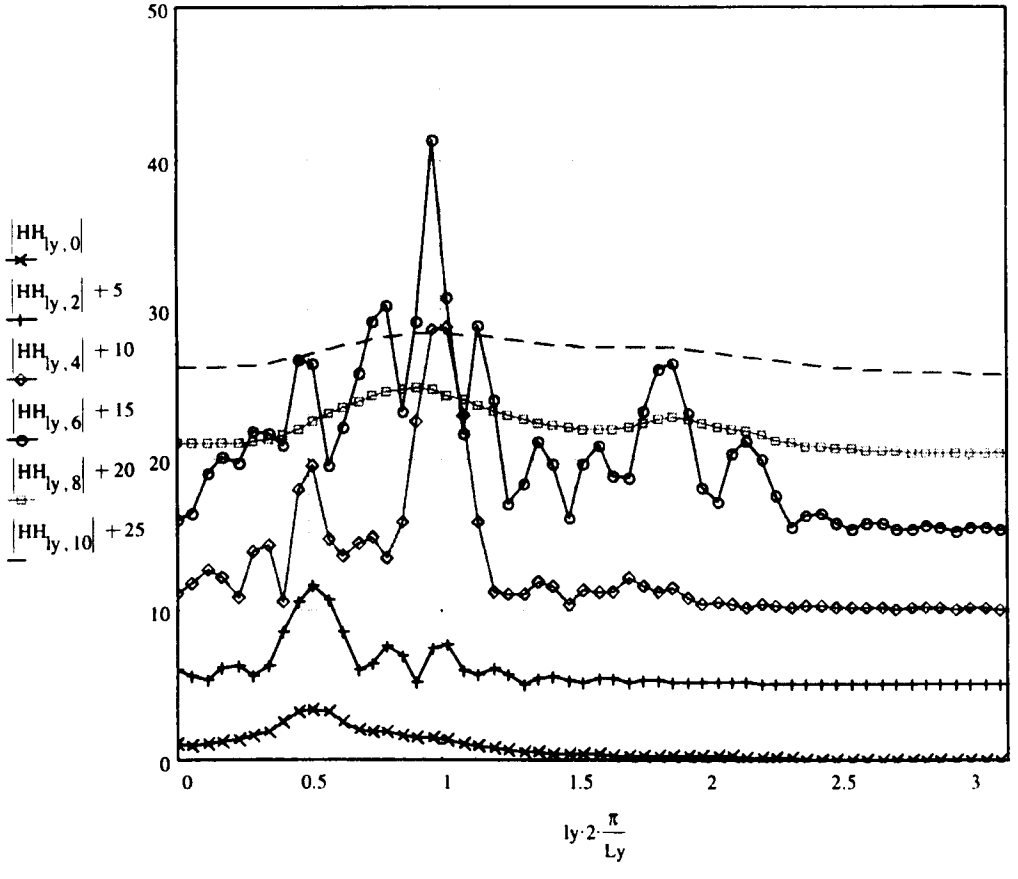


Fig.2.19. The chirp z-transform is for this purpose reduced to a FFT not only on the unit circle in the z plane but on a number of concentric circles with radii 1, 0.9, 0.8, 0.7, 0.6, 0.5. The bottom functional display is for $\rho = 1$, the upper one for $\rho = .5$.

2.15. Conclusions

The chirp z transform solution in the non-stationary deconvolution problem is only a first approach to a finer analysis. The idea is, once the poles are roughly located to concentrate more into details on those regions in the z plane and then to locally analyse them with chirp z-transforms. However, as the apparent usefulness of such an analysis is not very great no further efforts were made to fine tune the analysis.

A 2D analysis is also possible with the same approach but was not realised. The problem becomes much more complicated because in 2D one has to look for zero planes in a 4D space instead of zero points just in a plane. As we were looking for 2D techniques for deconvolution the idea was abandoned.

CHAPTER 3

Algebraic Approach to Non-Stationary Convolution and Deconvolution

3.1. Introduction

So far in this thesis, signals and systems were only treated in a very general way. As we will evolve more and more to 2D systems, some definition of equivalent terminologies are appropriate. Optics was involved with images long before they could be digitised. A lot of terminology comes from this field and is still in use :

- The input signal is very often called the **object function** in the object plane.
- The impulse response in a 1D system is translated for 2D as an **instrument function or point spread function (psf).** Because of the convolution effect one also speaks of a **blurring function**.
- The output signal is named the **data function** in the image plane.

The algebraic solution of 1-D convolution/deconvolution seems at a first glance rather simple. Convolution is straightforward. Equation (2.5) tells us that convolution, stationary or non-stationary, results in a multiplication of a matrix H and a vector x :

$$y = H \,.\, x \quad \text{and} \quad x = H^{-1} \,.\, y \tag{3.1}$$

The same expression is valid for a 2D system.

Deconvolving means, given the data function y and the instrumentation function H find the object function x by solving a linear system of algebraic equations. In simple terms we deblur the image. Most of the work in the deconvolution operation turn out to be the inversion of matrix H . There are no problem with this unless :

(i) H is singular and the inversion is not possible. For stationary processes in 1D : $Y(z) = H(z).\ X(z)$. Inverting the process results in $X(z) = Y(z)/H(z)$. This shows that if H(z) has zeros the operation in the frequency domain is not possible. The matrix method suffers from similar problems with H^{-1}, whether it is stationary or non-stationary.

(ii) H is a circulant or block circulant matrix depending on whether the origin of the signal is one or two dimensional. Certainly for 2-D signals the H matrices can become very large and consequently the inversion a very (computer) time consuming operation. For stationary processes the H matrices can via the FFT be converted into a diagonal or block diagonal matrix; these transformed matrices can very easily be inverted. Normally there is no need to do so because the transfer function method is much more efficient.

(iii) For non-stationary processes the classical convolution property does not hold anymore, in this manner only the matrix method is applicable. In this respect two algorithms have been investigated namely an under-relaxation scheme and a conjugate gradient method.

3.2. Non Stationary Convolution

In Chapter 2 a preliminary definition of non-stationary convolution was given. We now investigate the problem in greater depth. In the continuous case, convolution is an integral operation that associates the object function x to the data function y via a given point spread function h. Non-stationary convolution arises whenever the instrument function is not identical for all points over which the object function is convolved. If, for example, the convolution models the blurring of an image, the non-stationary form will result in portions of the image being less resolved (more blurred) than others.

Several types of non-stationary processes can occur :

1. Amplitude variation : The frequency content of the instrument function stays the same but the amplitude changes.

2. Frequency variation : The amplitude of the instrument function stays the same but the spectrum changes.
3. Energy conservation : Combination of both types mentioned above but with constant energy in the instrument function.
4. Shape variation : The most general case when the fundamental shape of the instrument function varies.

To model non-stationary convolution, the stationary convolution operation can be extended from

$$y(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau \qquad [\text{ or } \quad y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) \ ] \tag{3.2}$$

to : $$y(t) = \int_{-\infty}^{\infty} h(\tau, t-\tau)x(t-\tau)d\tau \quad \text{or} \quad [\ y(n) = \sum_{k=-\infty}^{\infty} h(k, n-k)x(n-k) \ ] \tag{3.3}$$

The impulse response h(t) is variable and changes with the delay (t-τ).

In 2D the discrete convolution is modified from

$$y(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k=-\infty_2}^{\infty} h(k_1, k_2)x(n_1 - k_1, n_2 - k_2) \tag{3.4}$$

to : $$y(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k=-\infty_2}^{\infty} h(k_1; n_1 - k_1, k_2; n_2 - k_2)x(n_1 - k_1, n_2 - k_2) \tag{3.5}$$

The discrete instrument function is variable and changes with displacement (n_1-k_1,n_2-k_2). For practical cases the boundaries of summation are not infinite because the convolution masks are limited to a length of K (1D and 2D case) and a width of L (2D case). Expression (3.3) and (3.5) becomes.

$$y(n) = \sum_{k=-0}^{K-1} h(k, n-k)x(n-k) \tag{3.6}$$

and $$y(n_1,n_2) = \sum_{k_1=-0}^{K-1} \sum_{k_2=0}^{L-1} h(k_1;n_1-k_1,k_2;n_2-k_2)x(n_1-k_1,n_2-k_2) \qquad (3.7)$$

3.3. The Convolution as an Algebraic Operation

Chapter 2 described the procedure for constructing circulant matrices. To extend them to non-stationary circulant matrices one fills in every column with the appropriate impulse response corresponding to that delay. In images one says that every point of the object function has a different point spread function and convolution with these point spread functions has to be accomplished. A variable blurring effect all over the picture is the result.

Some examples will be used to describe the construction of circulant matrices for stationary and non-stationary signals. First we consider a stationary signal composed of 2 short impulse responses concatenated one after the other. { $[\alpha_n]$ and $[\beta_n]$ stand in fact for $[\alpha^n]$ and $[\beta^n]$ with $n \in 0...3$ }.

h1 = [$\alpha_0\ \alpha_1\ \alpha_2\ \alpha_3$] and h2 = [$\beta_0\ \beta_1\ \beta_2\ \beta_3$] $\rightarrow$ H1 : stationary matrix

$$H1 = \begin{bmatrix} \alpha_0 & \beta_3 & \beta_2 & \beta_1 & \beta_0 & \alpha_3 & \alpha_2 & \alpha_1 \\ \alpha_1 & \alpha_0 & \beta_3 & \beta_2 & \beta_1 & \beta_0 & \alpha_3 & \alpha_2 \\ \alpha_2 & \alpha_1 & \alpha_0 & \beta_3 & \beta_2 & \beta_1 & \beta_0 & \alpha_3 \\ \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & \beta_3 & \beta_2 & \beta_1 & \beta_0 \\ \beta_0 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & \beta_3 & \beta_2 & \beta_1 \\ \beta_1 & \beta_0 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & \beta_3 & \beta_2 \\ \beta_2 & \beta_1 & \beta_0 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & \beta_3 \\ \beta_3 & \beta_2 & \beta_1 & \beta_0 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 \end{bmatrix}$$

The non-stationary version H2 will be constructed with the same 2 impulse responses but now in a delayed construction . The first four columns are constructed

with samples of the first impulse response, from the 5^{th} column on samples of the second impulse response are used.

$$H2 = \begin{bmatrix} \alpha_0 & 0 & 0 & 0 & 0 & \beta_3 & \beta_2 & \beta_1 \\ \alpha_1 & \alpha_0 & 0 & 0 & 0 & 0 & \beta_3 & \beta_2 \\ \alpha_2 & \alpha_1 & \alpha_0 & 0 & 0 & 0 & 0 & \beta_3 \\ \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & 0 & 0 & 0 & 0 \\ 0 & \alpha_3 & \alpha_2 & \alpha_1 & \beta_0 & 0 & 0 & 0 \\ 0 & 0 & \alpha_3 & \alpha_2 & \beta_1 & \beta_0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_3 & \beta_2 & \beta_1 & \beta_0 & 0 \\ 0 & 0 & 0 & 0 & \beta_3 & \beta_2 & \beta_1 & \beta_0 \end{bmatrix} \quad : \text{ stationary matrix}$$

The multiplication's of H1 with the vectors $x1 = [1\ 1\ 1\ 1\ 0\ 0\ 0\ 0]^T$ and $x2 = [0\ 0\ 0\ 0\ 1\ 1\ 1\ 1]^T$ are equivalent to the convolutions $y1 = h1 * x1$ and $y2 = h1 * x2$. Note how the step response function changes due to the delay incorporated in x2 compared with x1 :

$$\begin{bmatrix} \alpha_0 & 0 & 0 & 0 & 0 & \beta_3 & \beta_2 & \beta_1 \\ \alpha_1 & \alpha_0 & 0 & 0 & 0 & 0 & \beta_3 & \beta_2 \\ \alpha_2 & \alpha_1 & \alpha_0 & 0 & 0 & 0 & 0 & \beta_3 \\ \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & 0 & 0 & 0 & 0 \\ 0 & \alpha_3 & \alpha_2 & \alpha_1 & \beta_0 & 0 & 0 & 0 \\ 0 & 0 & \alpha_3 & \alpha_2 & \beta_1 & \beta_0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_3 & \beta_2 & \beta_1 & \beta_0 & 0 \\ 0 & 0 & 0 & 0 & \beta_3 & \beta_2 & \beta_1 & \beta_0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} \alpha_0 \\ \alpha_1 + \alpha_0 \\ \alpha_2 + \alpha_1 + \alpha_0 \\ \alpha_3 + \alpha_2 + \alpha_1 + \alpha_0 \\ \alpha_3 + \alpha_2 + \alpha_1 \\ \alpha_3 + \alpha_2 \\ \alpha_3 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \alpha_0 & 0 & 0 & 0 & 0 & \beta_3 & \beta_2 & \beta_1 \\ \alpha_1 & \alpha_0 & 0 & 0 & 0 & 0 & \beta_3 & \beta_2 \\ \alpha_2 & \alpha_1 & \alpha_0 & 0 & 0 & 0 & 0 & \beta_3 \\ \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & 0 & 0 & 0 & 0 \\ 0 & \alpha_3 & \alpha_2 & \alpha_1 & \beta_0 & 0 & 0 & 0 \\ 0 & 0 & \alpha_3 & \alpha_2 & \beta_1 & \beta_0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_3 & \beta_2 & \beta_1 & \beta_0 & 0 \\ 0 & 0 & 0 & 0 & \beta_3 & \beta_2 & \beta_1 & \beta_0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} \beta_3+\beta_2+\beta_1 \\ \beta_3+\beta_2 \\ \beta_3 \\ 0 \\ \beta_0 \\ \beta_1+\beta_0 \\ \beta_2+\beta_1+\beta_0 \\ \beta_3+\beta_2+\beta_1+\beta_0 \end{bmatrix}$$

Note also the delay and the circularity in the response ; β_0 shows up as the 5th sample, the first sample composed from $\beta_3 + \beta_2 + \beta_1$ should, in a non circular convolution, come after the 8th composed from $\beta_3 + \beta_2 + \beta_1 + \beta_0$ and so on.

In the next example the symbolic impulse responses are changed for more elaborate and numerically composed signals. Signals f1 and f2 composed of 64 samples is constructed. The first 32 of them are non-zero and are constructed from 2 sine waves f1 and f2 in different ways.

$N := 32 \qquad N1 := N - 1 \qquad N2 := \text{floor}\left(\frac{N1}{2}\right)$

$k := 0..N1 \qquad i := 0..N1 \qquad n := 0..2 \cdot N1$

$$f1_n := \sin\left(2 \cdot \pi \cdot n \cdot \frac{3}{8}\right) \qquad f2_n := 2 \cdot \sin\left(2 \cdot \pi \cdot n \cdot \frac{1}{8}\right)$$

f is composed of 2×16 samples of f1 and f2 put one after the other. ff is slightly different and consist in 32 samples of f1 and f2 added together. ff1 takes just 16 samples of f1 and ff2 takes 16 sample of f2.

$$f_n := if(n \le 32, if(n \le 16, f1_n, f2_n), 0)$$

$$ff_n := if(n \le 32, f1_n + f2_n, 0)$$

$$ff1_n := if(n \le 32, if(n \le 16, f1_n, 0), 0)$$

$$ff2_n := if(n \le 32, if(n \le 16, f2_n, 0), 0)$$

p1 and p2 are two pulses of respectively 16 and 32 sample wide. They must be seen as different windows to define the 'life time' of the signals f1 and f2. In f for example f1 and f2 are non-zero for only 16 samples. On the contrary in ff, they are non zero for the first 32 samples. This will have its implications on the spectra of f and ff . In fact, multiplication's in time domain of the signals and the windows result in convolution of their respective FT's. This could produce leakage in the resulting spectrum. An arbitrary combination of different windows starting and ending not at the same time, will produce an overall leakage and allow no detection of eventual starting time of signals. (which is too deeply hidden in the phase information)

$$p1_n := if(n \le 16, 1, 0) \qquad p1sh_k := p1_k \qquad P1 := cfft(p1sh)$$

$$p2_n := if(n \le 32, 1, 0) \qquad p2sh_k := p2_k \qquad P2 := cfft(p2sh)$$

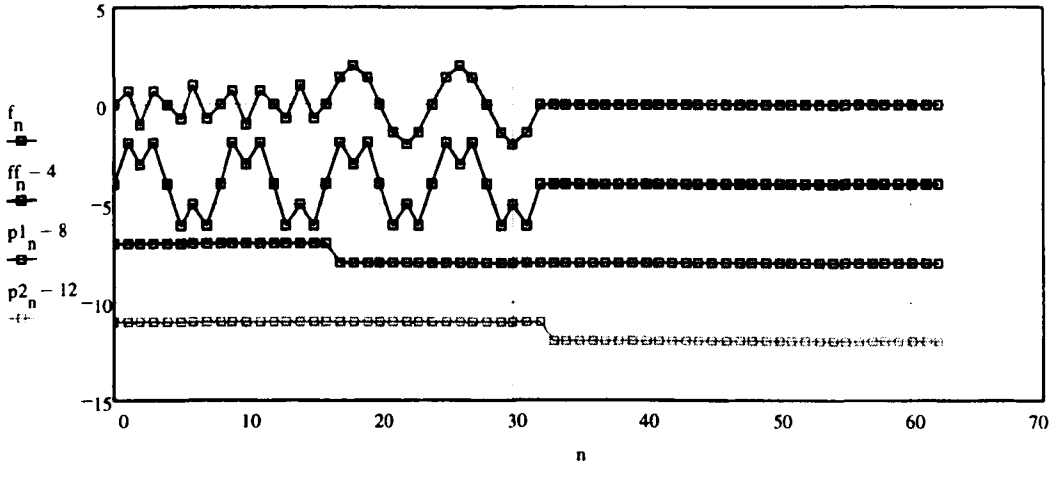


Fig.3.1. Two sine waves are composed in a different way and result in f and ff. The DFT of both signals will obscure their difference in time domain. Other techniques should be investigated to detect when signals start and stop.(See Chapter 5)

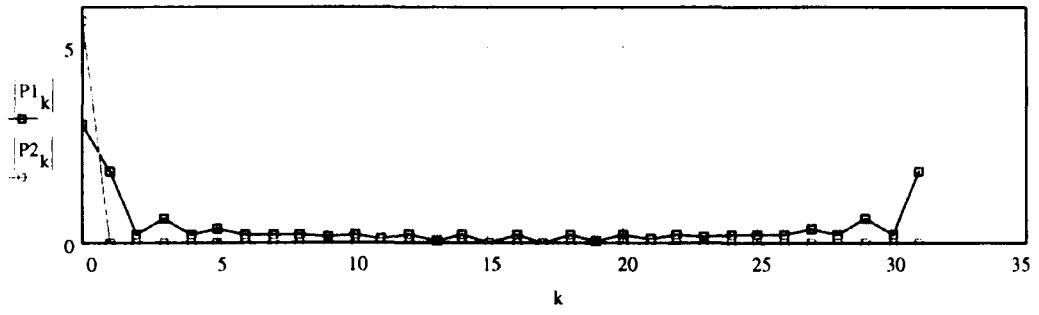


Fig.3.2. Amplitude spectrum of the window pulses p1 and p2. P1 is the spectrum for the 16 samples width pulse, P2 for the 32 samples width pulse. P1 will produce leakage in the spectrum analysis when considering an FT analysis of the first 32 samples.

Stationary circulant matrices for the first 32 samples will be constructed in F for f and in FF for ff. FF12 is a non-stationary construction with ff1 in the first part of the circulant matrix and ff2 in the second one.

$$W_{k,i} := e^{\frac{j\cdot 2\cdot\pi\cdot k\cdot i}{N}} \qquad F_{k,i} := if(k-i\geq 0, f_{k-i}, f_{N1+1+k-i})$$

$$FF_{k,i} := if(k-i\geq 0, ff_{k-i}, ff_{N1+1+k-i})$$

$$FF12_{k,i} := if(k-i\geq 0, if(i\leq N2, ff1_{k-i}, ff2_{k-i}), if(i\leq N2, ff1_{N+1+k-i}, ff2_{N+1+k-i}))$$

Diagonalization in (D, DD and DD12) produces the diagonal elements (r, rr and rr12)

$$D := W^{-1}\cdot F\cdot W \qquad DD := W^{-1}\cdot FF\cdot W \qquad DD12 := W^{-1}\cdot FF12\cdot W$$

$$r_k := D_{k,k} \qquad rr_k := DD_{k,k} \qquad rr12_k := DD12_{k,k}$$

The FFT's of the first 32 samples of f and ff , respectively called fsh(ort) and ffsh(ort) and abbreviated to fsh and ffsh produce

$fsh_k := f_k$ $\quad Fsh := cfft(fsh)$

$ffsh_k := ff_k$ $\quad FFsh := cfft(ffsh)$

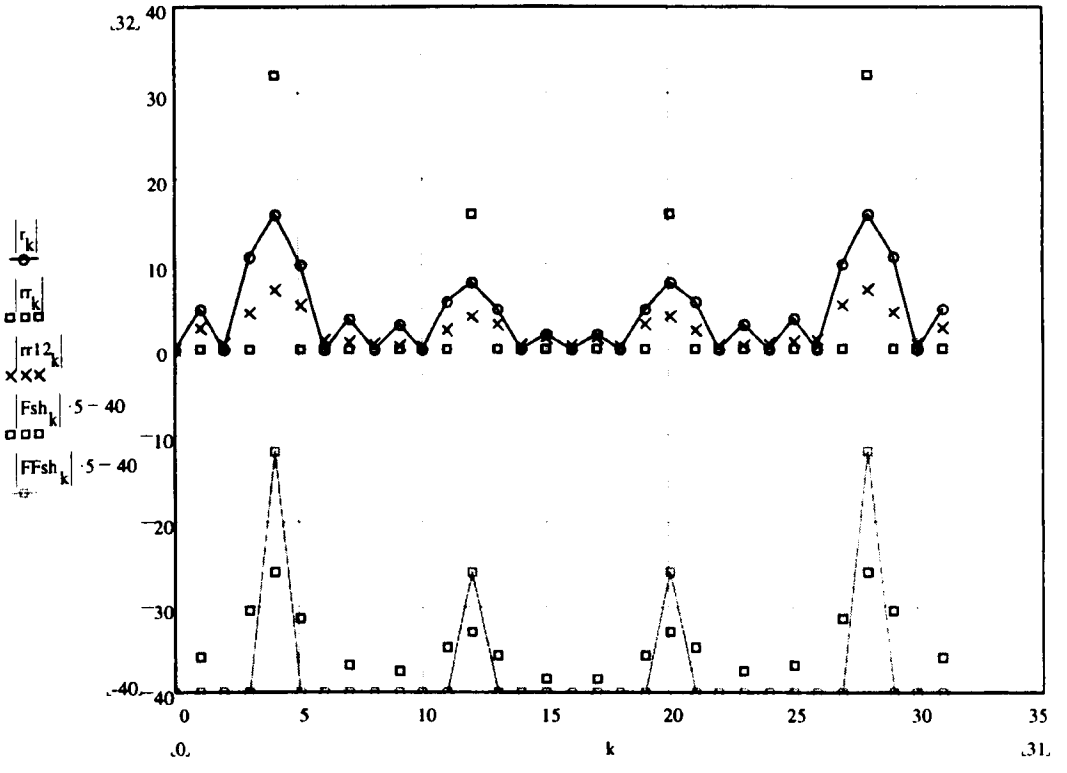


Fig.3.3. The upper part of the image shows the result of diagonalization . r shows the leakage in the spectrum . It could be seen as a warning that windowing was applied and the spectrum of those windows were convolved with the pure spectral lines. rr is perfect as spectrum information. The lower part shows the DFTs of f and ff. The results are similar with the upper part ones.

The picket fence effect in r is due to the fact that the signals f1 and f2 are not permanently present or commensurate in the 32 sampled time window. rr gives a perfect result as long as the time window is not extended. The 32 samples spectrum of f1 and f2 (FFsh) produces Fsh. In general one can conclude that Fourier techniques

whether using FT or diagolization produces errors from the moment non-stationarity shows up.

To get a better view on what is really happening, all elements with an absolute value negligible (>.1%) compared with the maximum value are made zero in the matrices. We consider successively D1, the manipulated matrix D which consists of the eigenvalues of f. D2 represents DD, containing the eigenvalues of ff and finally D3 the result of Fourier transform of a non-stationary signal. Remark the leakage in D1 compared with D2 and the results in D3 which is not diagonal at all. Although the greatest elements are in the neighbourhood of the diagonal , taking only those values into account results in loss of information about the frequency content of the signal.

Conclusion : Lossless non-stationary deconvolution in frequency domain is not possible. Other techniques (See next section and Chapter 5) must be used to deconvolve non-stationary signals.

$$D1_{k,i} := \text{if}\left[\left|D_{k,i}\right| \le \frac{\left|\max(D)\right|}{1000}, 0, \left|(D)_{k,i}\right| \right]$$

D1 =

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	4.748	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	10.912	0	0	0	0	0
4	0	0	0	0	16	0	0	0	0
5	0	0	0	0	0	9.892	0	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	3.546	0
8	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0

The diagonal values are proportional to the amplitude values of r_k in Figure 3.3

$$D2_{k,i} := \text{if}\left[\left|DD_{k,i}\right| \le \frac{\left|\max(DD)\right|}{1000}, 0, \left|(DD)_{k,i}\right| \right]$$

D2 =

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	32	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0

The value 33 stands for the amplitude part of f2 in ff (value rr_5 in Fig.3.3)

$$D3_{k,i} := \text{if}\left[\left|DD12_{k,i}\right| \le \frac{\left|\max(DD12)\right|}{1000}, 0, \left|(DD12)_{k,i}\right| \right]$$

D3 =

	0	1	2	3	4	5	6	7
0	0	0.161	0	0.355	0.5	0.292	0	0.086
1	1.507	2.597	1.507	0.449	0.223	0.398	0.313	0.123
2	0	0.161	0.636	1.052	0.981	0.474	0	0.116
3	1.124	0.449	3.329	4.402	1.394	1.228	1.124	0.405
4	0.5	1.644	0.981	5.761	7.123	3.32	0.981	1.412
5	0.579	0.398	0.94	1.228	4.732	5.12	2.783	0.433
6	0	0.034	0	0.355	0.981	1.403	1.211	0.559
7	0.137	0.123	0.185	0.405	0.958	0.433	0.888	0.928
8	0	0.025	0	0.219	0.5	0.474	0	0.559
9	0.087	0.071	0.106	0.245	0.549	0.27	0.232	0.098
10	0	0.02	0	0.163	0.344	0.292	0	0.189
11	0.175	0.05	0.2	0.179	0.539	0.204	0.327	0.075
12	0	0.266	0	0.2	0.271	0.594	0	0.624
13	0.183	0.04	0.199	0.145	0.105	0.169	0.277	0.064

3.4. Computer implementation

For the 1D case there is no problem to realise a non-stationary deconvolution . The convolution matrices are not too large to be efficiently manipulated. For a 128 × 128 image the convolution matrix has 268,435,456 entries. Clearly, numerical stability as well as computing time are important challenges to overcome if algebraic deconvolution is to be of some value.

The proposed solutions relies on three facts :

1. Convolution matrices are often sparse, provided that the convolution masks are not too large compared to the support of the object function.
2. Convolution matrices are very structured : the position of all non-zero entries can be predicted from the PSF.
3. Point spread functions tend to have higher values at their origin, since the influence of the point where they are applied is nearly always predominant, hence the highest entries in the matrices will usually be found near the diagonals.(See examples)

This suggests that the best way to approach the deconvolution problem is to use iterative methods.

Two algorithms are particularly efficient :

- An under-relaxation scheme.
- A conjugate gradient method.

Blackledge et al. [1] found also that :

(i) Iterative methods applied to stationary processes are only marginally inferior to the usual restoration techniques (Wiener filter , etc..) especially for broad convolution kernels. Furthermore, signals restored using algebraic methods are qualitatively comparable to those obtained via Fourier transforms.

(ii) Iterative methods are more efficient than segmentation for non-stationary deconvolution.

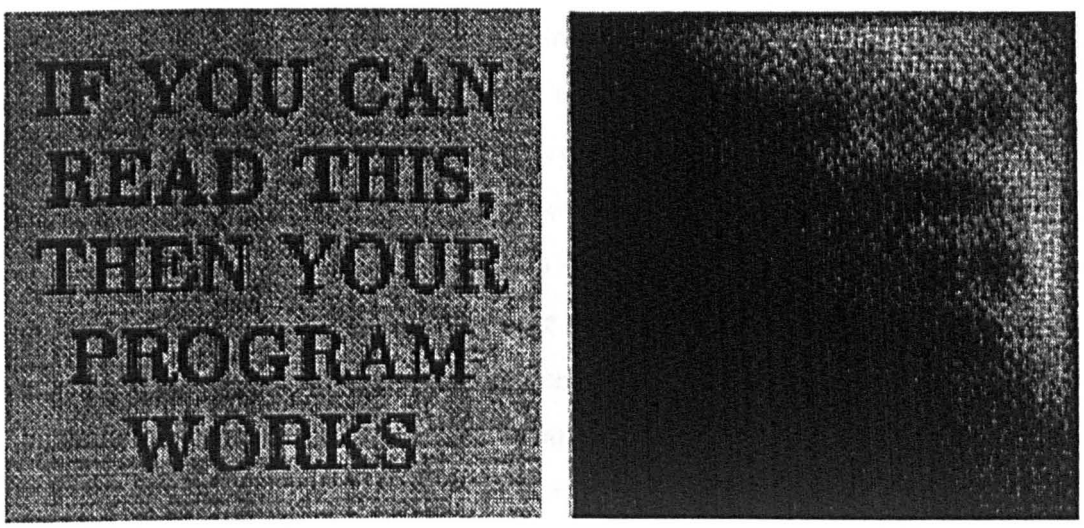
(iii) The best iterative method depends on the type of convolution kernel : If it is smooth and wide (Gaussian kernel) under-relaxation is more suitable; 32 iterations

typically produce a good quality restoration. Conversely, if a discontinuous and narrow kernel is used the conjugate method will produce more accurate and faster results. Here again, 32 iterations are often sufficient to obtain good quality restoration.

(iv) The main drawback of algebraic methods besides the computing and storage requirements, is the ill conditioned characteristics of the matrices. A contradiction arises : it is important that in ill-conditioned systems the number of iterations is kept minimal, however the more blurring in the data, the more iterations are required.

3.5. **Some Results**

Courtesy to Prof. J. Blackledge et al. , we now reproduce some results of the conjugate gradient method.



(a) (b)

Fig.3.4. (a) : 128 × 128 test picture. (b) : Non-stationary convolution for a tophat kernel with energy conservation (the average tophat half-width is $\lambda = 4$ pixels)

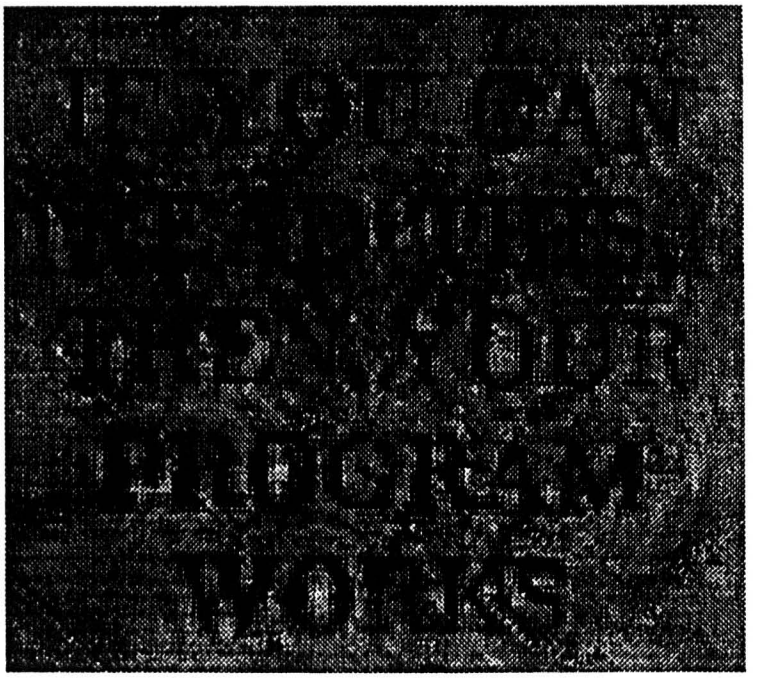


Fig.3.4.c. Algebraic restoration with the conjugate gradient method

CHAPTER 4

Digital Filters

4.1. Introduction

Chapter 2 covered some tools to describe and manipulate data. 1D and 2D convolutions on signals and images with respectively stationary impulse responses h(n) and point spread functions (psf) $h(n_1,n_2)$ resulted in a filtering operation like for example a blurring effect on an object function. The result of filtering can be very effectively studied in the frequency domain by analysing the z transforms H(z) and $H(z_1,z_2)$ of respectively h(n) and $h(n_1,n_2)$. When the signal or image is non-stationary the convolution can still take place but the z transform of the signal or image does not exist and the convolution theorem cannot be applied any more.

While filtering is a issue that can be described in the frequency domain by considering the positions of poles and zeros of the transfer function in z or (z_1,z_2) , their impulse response or psf can very easily be found by inverse z transforming. With zeros only, a convolution operation does the filtering. The convolution kernels will be called **Finite Impulse Response (FIR)** filters . When poles and zeros are involved a difference equation will represent the system in time domain (for 2D in the space domain) and will act upon an input signal to produce an output signals. This operation is called **Infinite Impulse Response (IIR)** filtering. Because of the poles these systems are not always stable and certainly for 2D there are no simple design rules as yet. Hence we will restrict ourselves in this chapter to FIR filters for the 2D case.

Non-stationary convolution and deconvolution has been studied in terms of matrix operations. (Chapter 3). All transforms so far were based on correlations between complex exponential functions existing between $-\infty$ and $+\infty$, and the

analysing signal or image. If the signal or images contains **local** information (in time or space) the Fourier transform will only produce **global** information in the frequency domain. One way to still use these global techniques is to split up a non-stationary signal or image into quasi stationary time or space slots and then use the classical analysing techniques. For speech analysis, for example, time slots of about 30 ms are used.

With this idea in mind it is still interesting to study the design of stationary filters. Here we consider the classical window approach and the Wiener filter technique and make a comparison .

4.2. FIR Filter Design

The duality principle discussed in Chapter 2 proved to be very useful to construct "brickwall" frequency characteristics. As filters are not the main topic of this thesis, only **symmetrical, odd numbered** discrete impulse responses are considered to realise linear phase filters. For a more elaborated discussion see [64]

4.2.1. Low Pass FIR Filter Design

The expression of the impulse response for a prototype low pass brickwall filter is :

$$h(n) = (d\_c).\mathrm{sinc}(n.d\_c) \qquad (4.1)$$

with $\mathrm{sinc}(n) = \sin(\pi n)/\pi n$ and d_c being equal to the duty cycle of a brickwall pulse. We have to define here the bandwidth of the filter. With for instance d_c = ¼ Fig.4.1 shows the impulse response of an FIR filter with a spectral response composed of a pulse with duty cycle = ¼.

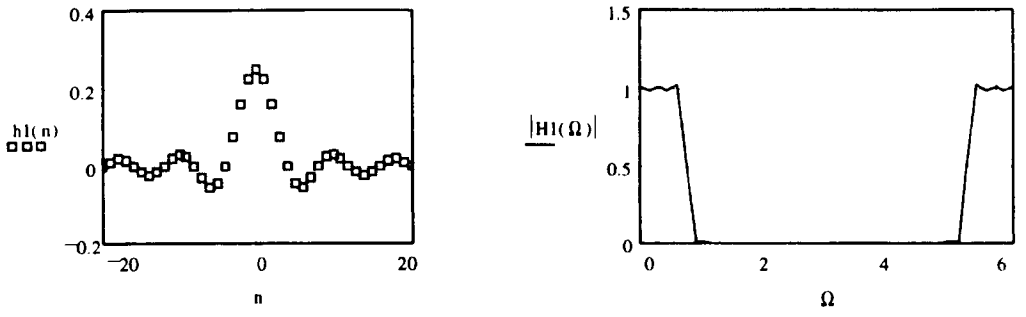


Fig. 4.1. h1(n) is the impulse response of a low pass FIR filter with a bandwidth of π/4.

To generalise the rule :

1. Choose a low pass frequency f_{lp} relative to half the sampling frequency $f_s/2$.
2. Make $d\_c = 2f_{lp}/f_s$.
3. Insert the value into Eq. (4.1) :

$$h_{lp}(n) = (2.f_{lp}/f_s).\mathrm{sinc}(n.2.f_{lp}/f_s) \qquad (4.2)$$

4. The resulting infinite expression for h(n) will have to be abbreviated to a finite length; hence the expression finite impulse response. This shortening will of course produce errors. They will be minimised by multiplying the impulse responses with appropriate windows (See section on windows).

4.2.2. High Pass FIR Filter Design

A high pass filter design makes use of the modulation principle : One can shift the frequency band to for example half sampling frequency $\Omega = \pi$ by multiplying the low pass filter construction with the cosine of this half sampling frequency. Suppose we try it with the low pass prototype filter designed in section 4.2.1.

The example will have a high pass characteristic starting at $3\pi/4$.

$$h(n) = (-1)^n.\ (d\_c).\mathrm{sinc}(n.d\_c) \qquad (4.3)$$

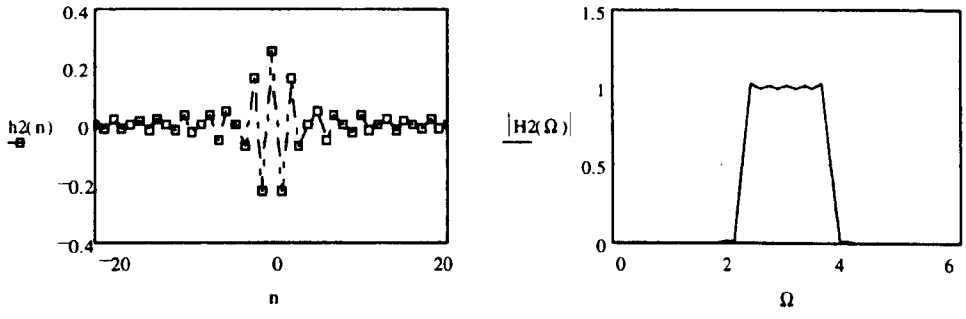


Fig.4.3. Modulating the low pass impulse response with cos(n.π) results in a high pass equivalent. f_{hp}= 3π/4

Design rules :

1. Choose a high pass frequency f_{hp} relative to half the sampling frequency $f_s/2$.
2. Calculate $f_2 = f_s/2 - f_{hp}$
3. Make $d\_c = 2f_2/f_s = 1-2f_{hp}/f_s$.
4. Insert the value in Eq. (4.3) :

$$h_{hp}(n) = (-1)^n.\,(1-2.f_{hp}/f_s).\mathrm{sinc}(n.(1-2.f_{hp}/f_s)) \qquad (4.4)$$

5. Make the expression finite by using windows.

4.2.3. Band Pass FIR Filter Design

A band pass FIR filter can be designed following the same procedure as that used to design the high pass filter. The low pass prototype is modulated at another central frequency Ω_{centre} by multiplying (4.1) with $\cos(2.\pi.n.f_{centre}/f_s)$.

The example will have a centre frequency of π/2 and a bandwidth of 2×π/4. The number of samples involved is 201 (from -100...100).

$n := -100, -99 .. 100$

$\Omega := 0, \frac{\pi}{20} .. 2 \cdot \pi$

$d\_c := .5$: duty cycle

$$h3(n) := \text{if}\left(n = 0, d\_c, d\_c \cdot \frac{\sin\left(\pi \cdot n \cdot \frac{d\_c}{2}\right)}{\pi \cdot n \cdot \frac{d\_c}{2}} \cdot \cos\left(n \cdot \frac{\pi}{2}\right)\right)$$

$$H3(\Omega) := \sum_{n=-\infty}^{\infty} h3(n) \cdot e^{j \cdot n \cdot \Omega}$$

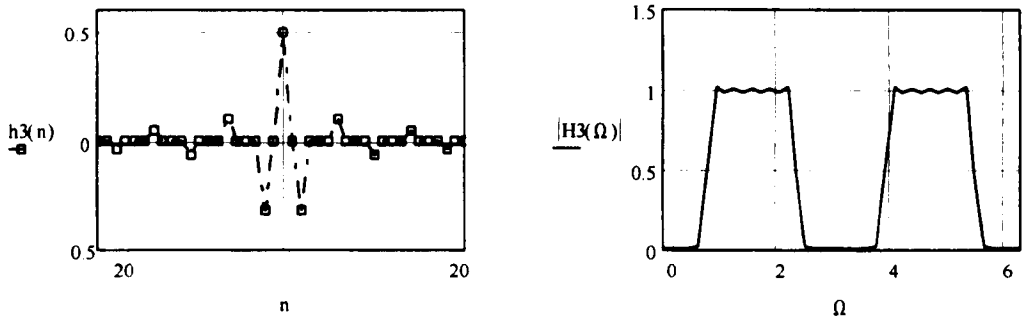


Fig.4.4. Impulse response and frequency characteristic of a bandpass filter with centre frequency π/2 and bandwidth 2×π/4.

Design rules :

1. Choose the width of the band pass filter f_{bw} and the centre frequency f_{centre} relative to half the sampling frequency $f_s/2$.
2. Substitute $d\_c = 2f_{bw}/f_s$.
3. Insert the d_c value in Eq. (4.1) . Inverse Fourier transforming proves that for band pass filters only half the d_c value should be put in the sinc function. Finally a multiplication with $\cos(\pi.n.f_{centre}/f_s)$ is necessary for the frequency shift :

$$h_{bp}(n) = 2.f_{bw}/fs \ .\mathrm{sinc}(n.f_{bw}/f_s).\cos(2\pi.n.f_{centre}/f_s) \qquad (4.5)$$

4. Make the expression finite by using windows.

4.2.4. Band Reject FIR Filter Design

Band reject filters can be designed starting from a band pass design. A band reject characteristic results from the subtraction of a constant frequency characteristic minus the characteristic of a band pass. Because z transforms are linear, one can transform back the subtraction operation to time domain :

$$1 - H_{bp}(z) \rightarrow \delta(n) - h_{bp}(n) \qquad (4.6)$$

The example will have a centre frequency of $\pi/2$ and a bandwidth of $2\times\pi/4$.

$n \quad 100,\ 99..\ 100$

$\Omega \quad 0, \frac{\pi}{20}\ ..\ 2\cdot\pi$

$d\_c \quad .5$: duty cycle

$$h4(n) \quad \text{if } n=0,\ 1 \quad d\_c,\ d\_c\cdot \frac{\sin\left(\pi\cdot n\cdot \frac{d\_c}{2}\right)}{\pi\cdot n\cdot \frac{d\_c}{2}}\cdot\cos\left(n\cdot\frac{\pi}{2}\right)$$

$$H4(\Omega) \quad \sum_{n=-\infty}^{\infty} h4(n)\cdot e^{j\cdot n\cdot\Omega}$$

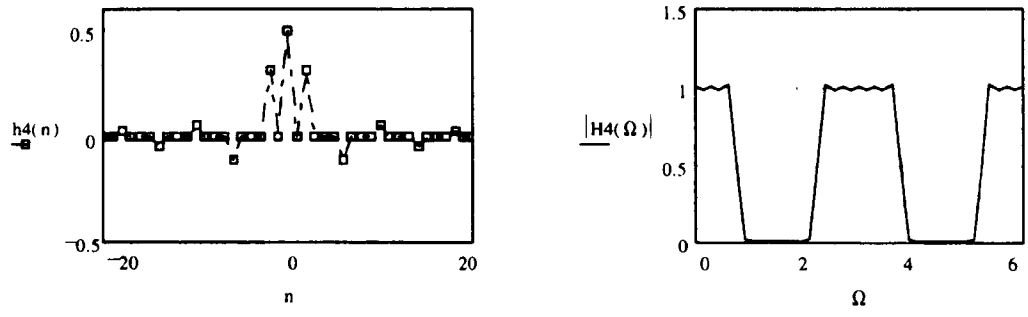


Fig.4.4. Impulse response and frequency characteristic of band reject filter with centre frequency π/2 and bandwidth 2×π/4.

Design rules :

1. Choose the width of the band reject filter f_{br} and the centre frequency of the filter f_{centre} relative to half the sampling frequency $f_s/2$.
2. Substitute $d\_c = 2f_{br}/f_s$.
3. Insert the d_c value in Eq. (4.1) . Inverse Fourier transforming proves that for band pass filters only half the d_c value should be put into the sinc function. Finally a multiplication with $\cos(\pi.n.f_{centre}/f_s)$ is necessary for the frequency shift :

4. Apply Eq. (4.6).

$$h_{br}(n) = 1 - 2.f_{br}/fs\ .\mathrm{sinc}(n.f_{br}/f_s)).\cos(2\pi.n.f_{centre}/f_s) \qquad (4.7)$$

5. Make the expression finite by using windows.

4.3. Windows

Due to the necessary truncation of the impulse responses their values abruptly drop to zero. One can consider this operation in terms of the multiplication of the infinite impulse response with a finite rectangular window. Due to the modulation

property, this multiplication in the time domain will result in a convolution in the frequency domain of the individually transformed signals. The rectangular window produces a sinc function in the frequency domain. The convolution of this sinc function with the ideal brickwall structure (flat characteristics in pass and stop bands and an infinite steep transient region) results in a much less ideal characteristic with an oscillatory pass and stop band and a finite steep transient region. This is called the Gibb's phenomena (See ripple and transition band in figures in section 4.2).

To diminish these imperfections in the frequency domain a window with less side lobes in the frequency domain than that associated with a rectangular window could be used to multiply with the infinite impulse response. Hamming, Hanning and Kaiser are good windows to cope with this problem.

An example illustrates the effect of a Hanning window. Note the disappearance of the oscillation in the pass and the stop bands, the better attenuation in the stop band but, as a disadvantage, the less steep the characteristic in the transition band. Better results can be reached with more samples in the impulse response. Intensive study would lead us to far and the reader is referred for other design techniques as well to a more specialised literature to [62] . These design tools were presented here as a first approach to 2D filtering. 2D kernels for convolution [the point spread function $h(n_1,n_2)$] could very easily be constructed by designing a spatial sinc function out of the one used earlier. We see however, that a Bessel function gives better results for constructing 2D FIR filters.

$w \quad \text{hanning}(21) \qquad nl \quad 0..20 \qquad hl(nl) \quad h(nl \quad 10)$

$w(nl) \quad w_{nl} \qquad h_w(nl) \quad w(nl)\cdot hl(nl) \qquad H_w(\Omega) \quad \sum_{nl=0}^{20} h_w(nl)\cdot e^{j\cdot nl\cdot\Omega}$

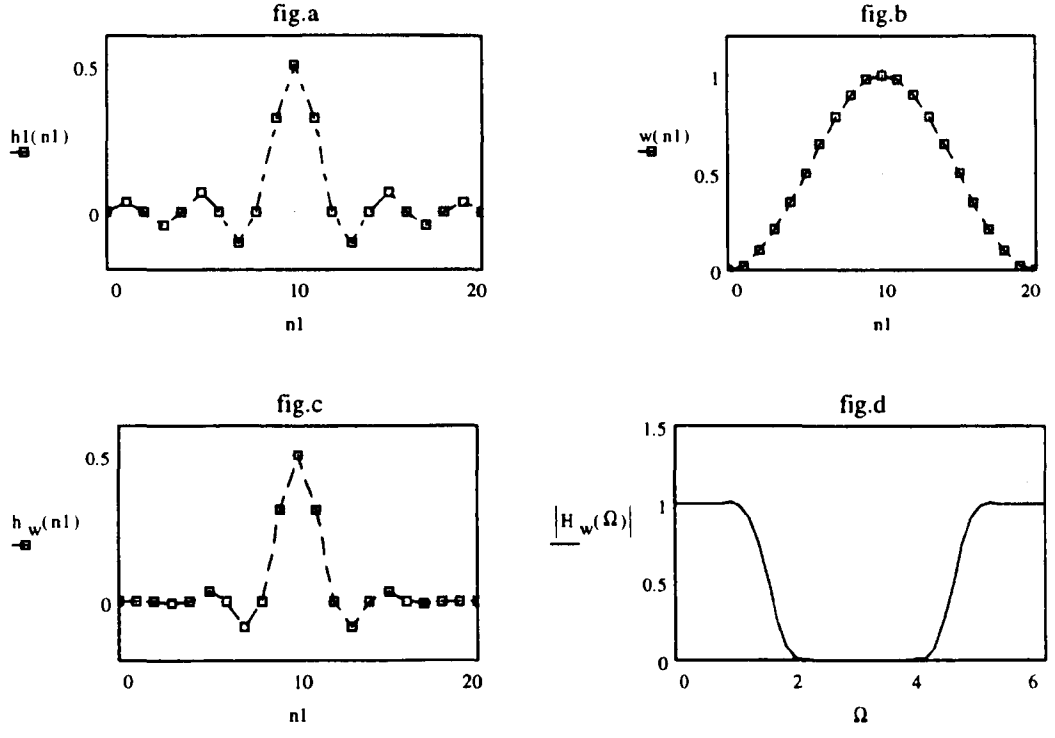


Fig.4.5. Fig.a shows a truncated sinc function . Fig.b represents the Hanning window. Note the Gibb's phenomena. Fig.c is the result of the truncated sinc function multiplied by a Hanning window. Fig.d presents a much smoother frequency characteristic,the disadvantage being the larger transition band.

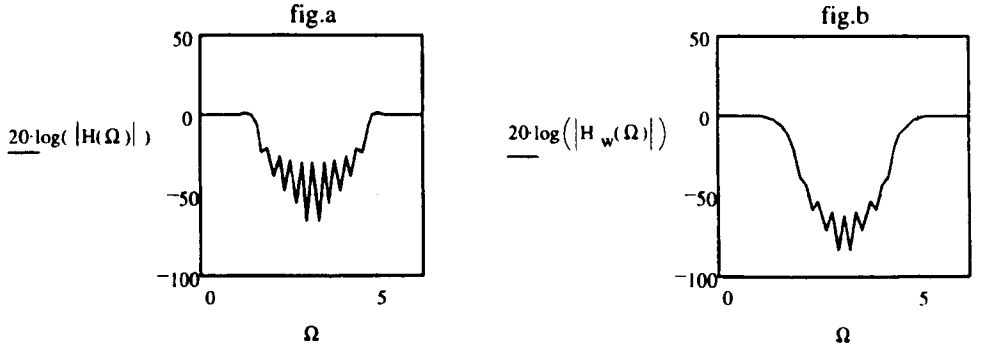


Fig.4.6. The logarithmic characteristic shows in fig.b a much larger attenuation then in fig.a. (Windowed filter)

4.4. 2D FIR Filter Design

We will only concentrate on the low pass and high pass design because they are most used in image processing. Low pass filtering is a primitive way of noise filtering because most of the significant frequency components in a signal are low frequencies. High frequency components with no valuable data can be omitted. Contrast changes can be detected in a simple way by performing high pass filtering.

4.4.1. 2D FIR Low Pass Filter

A first attempt to design a 2D FIR low pass filter can be by starting from a 1D windowed (Hanning) sinc function.

$N := 31$

$n1 := 0..N-1 \qquad n2 := 0..N-1 \qquad h_{n1,n2} :=$

$N1 \leftarrow 15$

$r \leftarrow \sqrt{(n1-N1)^2+(n2-N1)^2}$

$\delta \leftarrow 2$

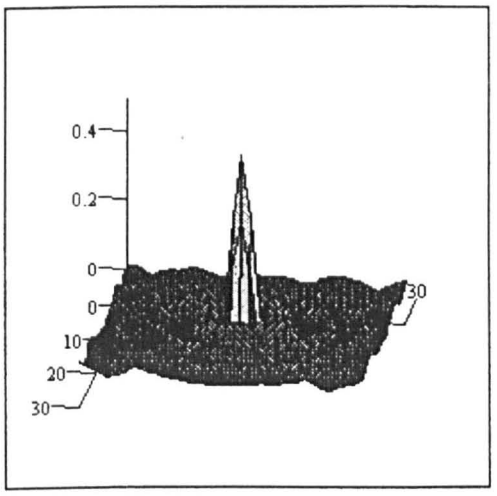
$T \leftarrow 4$

$$\text{if}\left[r=0, \frac{\delta}{T}, \frac{\delta}{T}\cdot\frac{\sin\left(\pi\cdot r\cdot\frac{\delta}{T}\right)}{\left(\pi\cdot r\cdot\frac{\delta}{T}\right)}\cdot\left[.5\cdot\left(1+\cos\left(2\cdot\pi\cdot\frac{r}{N1}\right)\right)\right]\right]$$

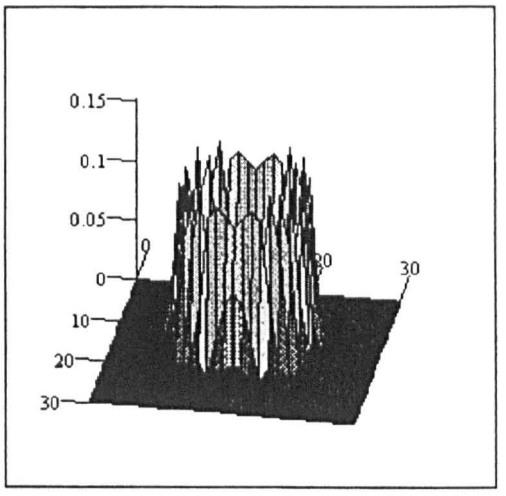
$$H := \text{cfft}(h)$$

$$H\_shift := \text{fft\_shift}(H)$$

$$H\_amp_{n1,n2} := \left|H\_shift_{n1,n2}\right|$$



h



H_amp

Fig.4.7. Spatial sinc function and its disappointing amplitude spectrum. Frequency 0 for both frequency axis is set in the middle of the frequency plane with a fft_shift operation.

Using a spatial sinc function produces too much ripple at the edge of the transition band. This is due to the fact that the inverse Fourier transform of a mesa-like spatial frequency representation (= 2D low pass frequency representation) results in a spatial Bessel function.[59]. To illustrate this consider the following example:

$n1 := -10.-9..10$

$n2 := -10.-9..10$

$R := \frac{\pi}{2}$

$$h_{lp}(n1,n2) := if\left[(n1=0)\cdot(n2=0), \frac{R^2}{\pi}, \frac{R}{2\cdot\pi\cdot\sqrt{n1^2+n2^2}}\cdot J1\left(R\cdot\sqrt{n1^2+n2^2}\right)\right]$$

To shift the design results from (-10..10) to (0...20) the following instructions should be made:

$k1 := 0..20$

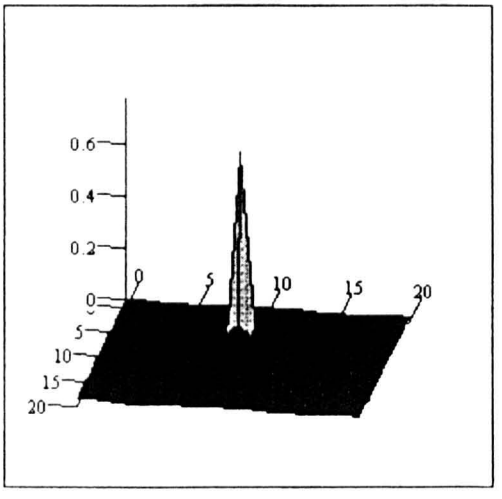
$k2 := 0..20$

$$hs\_lp_{k1,k2} := h\_lp(k1 - 10, k2 - 10)$$

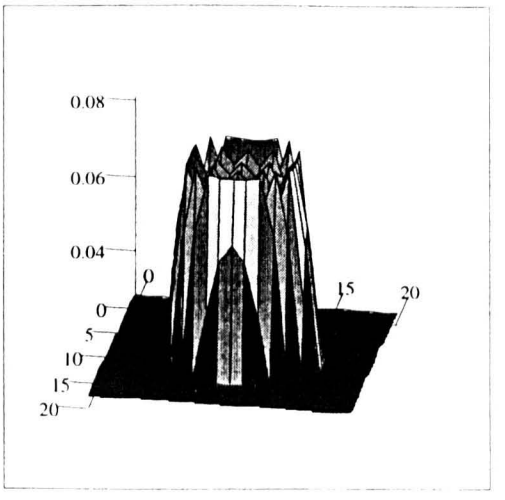
$$H\_lp := cfft(hs\_lp)$$

$$Hlp\_amp_{k1,k2} := |H\_lp_{k1,k2}|$$

$$Hslp\_amp := fft\_shift(Hlp\_amp)$$



hs_lp



Hslp_amp

Fig.4.8. A Bessel function of the first kind and the first order gives much better results than the sinc function. Windowing could be considered but is, due to the lack in accuracy in our visual system, not always necessary.

The design tools can be used to construct a low pass kernel of 7×7 with a cut off frequency of $\pi/3$. To keep the DC level filter invariant ,the point spread function should be normalised by the sum of all point spread function values. The resulting kernel is defined in $h1_{lp}$.

$n1 := -3, -2..3$

$n2 := 3, 2..3$

$R := \frac{\pi}{3}$

$$h_{lp}(n1,n2) := \text{if}\left((n1=0)\cdot(n2=0), \frac{R^2}{\pi}, \frac{R}{2\cdot\pi\cdot\sqrt{n1^2+n2^2}}\cdot J1\left(R\cdot\sqrt{n1^2+n2^2}\right)\right)$$

$k1 := 0..6$

$k2 := 0..6$

$$sum\_h\_lp := \sum_{n1}\sum_{n2} h_{lp}(n1,n2)$$

$$h1_{lp_{k1,k2}} := \frac{h_{lp}(k1-3,k2-3)}{sum\_h\_lp}$$

Before applying the filtering on an image let us first introduce the Kagray image and consider its spectrum in Fig.4.9.

$f1 := READBMP(kagray)$

$r\_f1 := rows(f1)$

$c\_f1 := cols(f1)$

$F1 := cfft(f1)$

$r1 := 0..r\_f1-1$

$c1 := 0..c\_f1-1$

$F1\_amp_{r1,c1} := |F1_{r1,c1}|$

$F1s\_amp := fft\_shift(F1\_amp)$

$F1log\_amp_{r1,c1} := \log\left(F1s\_amp_{r1,c1}\right)$

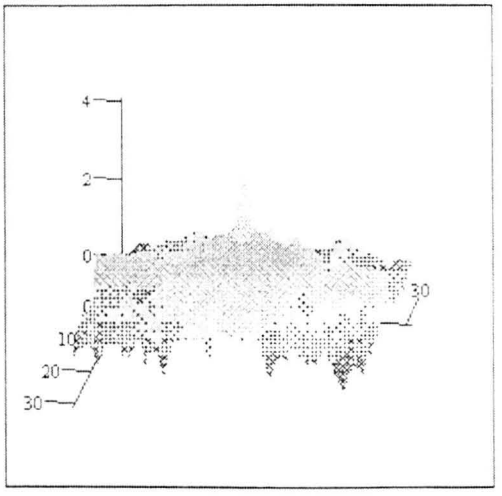


Fig.4.9. Kagray and its logarithmic amplitude spectrum. Note the absence in high frequency components in the picture.

An image is a non-stationary signal, this means in the wide sense that the probability density function (pdf) changes from place to place in the picture. A changing pdf also implies local frequency changes that can't be detected by the classical Fourier transform . It will only globally reveal the frequency content of the picture. Filtering based on stationary convolution does only a global filtering of the picture and local details just vanish under a low pass filter . To illustrate this let us consider a picture with noise. By eliminating the high frequency part of the noise the details in the picture will also be eliminated.

$$noise_{rl,cl} := rnd(31)$$

$$fl_{noise} := fl + noise$$

$$g_{lp} := twodconvol\left(fl_{noise}, hl_{lp}\right) \qquad : \text{ 2D convolution}$$

Fig.4.10. Kagray plus noise and its low pass filtered version. The noise is well eliminated but also the details are gone.

4.4.2. 2D FIR High Pass Filter

A high pass filter allows the extraction of details in a picture, but enhances noise (or the non significant details) as well. The design is very easy :

$$h_{hp}(n_1,n_2) = \delta(n_1,n_2) - h_{lp}(n_1,n_2) \qquad (4.8)$$

An example illustrates this. Let us consider a high pass filter of 21×21 elements which starts at

$f_{hp} = \pi/4.$

$n1 := -10, -9 .. 10$

$n2 := -10, -9 .. 10$

$R := \frac{\pi}{4}$

$$h_{hp}(n1, n2) := if\left[(n1=0)\cdot(n2=0), 1 - \frac{R^2}{\pi}, \left(\frac{R}{2\cdot\pi\cdot\sqrt{n1^2+n2^2}}\cdot J1\left(R\cdot\sqrt{n1^2+n2^2}\right)\right)\right]$$

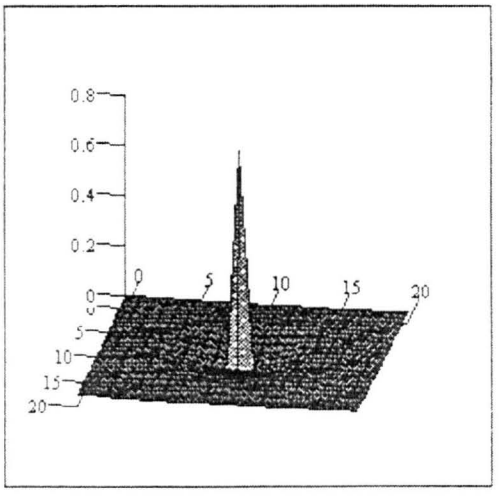
$k1 := 0 .. 20$

$k2 := 0 .. 20$

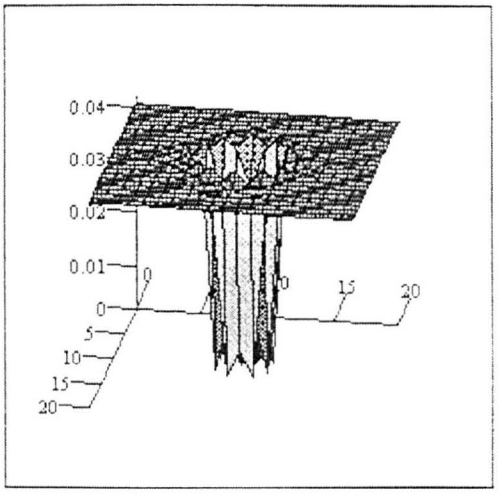
$$hl_{hp_{k1,k2}} := h_{hp}(k1-10, k2-10)$$

$$Hl_{hp} := cfft\left(hl_{hp}\right)$$

$$Hl\_amp\_hp_{k1,k2} := \left|Hl_{hp_{k1,k2}}\right| \qquad Hls\_amp\_hp := fft\_shift(Hl\_amp\_hp)$$



hl_{hp}



Hls_amp_hp

Fig.4.11. 2D High pass filter based on low pass Bessel function design and its frequency response.

For the image processing application we choose a 7×7 convolution kernel. First we consider filtering at $\Omega 1_{hp} = \pi/8$ and then at $\Omega 1_{hp} = \pi/6$. The higher the frequency, the smaller the edges because only the very fast contrast changes are captured by the filter.

Fig. 4.12. Kagray high pass filtering at $\Omega 1_{hp}= \pi/8$ and $\Omega 2_{hp}= \pi/6$. Raising the frequency cut off of the filter sharpens the edges where the contrasts changes

4.5. Optimal Linear Filter Design

4.5.1. Introduction

Wiener filters are optimal linear filters. The requirement for the filter is to make the expectation of the squared error "as small as possible" , i.e. to filter the image or data $x(n_1,n_2)$in such a way that it comes as close as possible to the original signal or object $s(n_1,n_2)$. Note that the original signal is blurred by a convolution with the point spread function $h(n_1,n_2)$ and additive noise(n_1,n_2). Very often the blurring effect is not considered in the Wiener filter design. In fact , if the blurring effect is known, inverse filtering can first take away this blurring effect and Wiener filtering can concentrate on removing noise. Special care must be taken in these cases so that the transfer function of the blurring effect shows no zeros in the working frequency band because

by inverse filtering these zeros becomes poles and create infinite values in the transfer function of the deblurred image. [56] Inverse filtering is not further studied here. We will only concentrate on the removal of noise in a signal or image. To start with the noise is considered stationary. The removal of non-stationary noise could, as suggested in the introduction, be attained by splitting up the signal or image into boxes (where the process is quasi stationary) doing a box-dependent filtering and then adding the pieces together. Of course some overlapping of the boxes should be considered otherwise annoying discontinuities would show up. [39]

Suppose a 1D or 2D signal is blurred by h in a convolution process and further deteriorated by noise. An FIR filter with coefficients w has to be designed to minimise the error.

$$x(n_1,n_2) = [s(n_1,n_2) * h(n_1,n_2)] + \text{noise} \rightarrow \underset{w(n_1,n_2)}{\text{Wiener filter}} \rightarrow s\_wien(n_1,n_2)$$

The error signal is $e(n_1,n_2) = s(n_1,n_2) - s\_wien(n_1,n_2)$

An expression like "as small as possible" means the minimisation of a cost function. For the Wiener filter design we choose for **the expectation of the mean square error as the cost function**.

Wiener filters are adaptive filters because their coefficients have to be recalculated for each new input. The filter is designed as an FIR filter and the operation is a 2D convolution of the input signal $x(n_1,n_2)$ with the Wiener filter coefficients. Stability is no problem because of the FIR structure.

4.5.2. Derivation of the Wiener Hopf Equation for 2D

Suppose that $x(n_1,n_2)$ is the noisy observation of the signal $s(n_1,n_2)$ blurred by $h(n_1,n_2)$ and further deteriorated with noise(n_1,n_2). The complete signal $x(n_1,n_2)$ must

be stationary . This procedure will minimise the mean square error between the original $s(n_1,n_2)$ and the processed $s\_wien(n_1,n_2)$ signal.

$$X(n_1,n_2) = [s(n_1,n_2) * h(n_1,n_2)] + noise \tag{4.9}$$

We wish to approximate $s(n_1,n_2)$ by applying a linear estimator $w(n_1,n_2)$ on $x(n_1,n_2)$:

$$s\_wien(n_1,n_2) = x(n_1,n_2) * w(n_1,n_2) \tag{4.10}$$

The error is defined as $e(n_1,n_2) = s(n_1,n_2) - s\_wien(n_1,n_2)$

or $$e(n_1,n_2) = s(n_1,n_2) - x(n_1,n_2) * w(n_1,n_2) \tag{4.11}$$

A cost function J is defined as the expectation of the mean square error

$$J = E[\, e(n_1,n_2)\, e(n_1,n_2)] \tag{4.12}$$

where E stands for the expectation operator.

The problem now is to determine the condition for which J has a minimum value. We will concentrate on real input data and coefficients because the image we consider is real valued. A 1-D complex analysis can be found in [69].

The filter coefficients are $w(n_1,n_2)$ $n1,n2 \in \mathbf{Z}$

Correspondingly, the gradient operator for the k^{th} coefficient is defined as :

$$\nabla_{k1,k2} = \partial/\partial w_{k1,k2} \tag{4.13}$$

By applying this gradient operator to the cost function J, a gradient matrix will be obtained with k_1,k_2 elements

$$\nabla J_{k1,k2} = \partial J/\partial w_{k1,k2} \tag{4.14}$$

To minimise the cost function, the gradient matrix must be simultaneously zero. Substituting equation (4.11) in (4.14) we obtain

$$\nabla J_{k1,k2} = E[2e(n_1,n_2)\, \partial e(n_1,n_2) / \partial w(k_1,k_2)\,] \qquad (4.15)$$

Using (4.9), (4.10) in (4.12) we have the following derivative

$$\partial e(n_1,n_2) / \partial w(k_1,k_2) = - x(n_1 - k_1, n_2 - k_2) \qquad (4.16)$$

Substituting this partial derivative into Eq. (4.15) and implementing the minimisation of the cost function leads to

$$E[e(n_1,n_2)\, x(n_1 - k_1, n_2 - k_2)\,] = 0 \qquad (4.17)$$

Substituting (4.11) in (4.17) gives

$$E\{\, s(n_1,n_2)\, x(n_1 - k_1, n_2 - k_2) - [\, w(n_1,n_2) * x(n_1,n_2)]\, x(n_1 - k_1, n_2 - k_2)\,\} = 0 \qquad (4.18)$$

or

$$E[\, s(n_1,n_2)\, x(n_1 - k_1, n_2 - k_2)] = w(n_1,n_2) * E[x(n_1,n_2)\, x(n_1 - k_1, n_2 - k_2)] \qquad (4.19)$$

The left hand side of equation (4.19) represents for the cross-correlation between $s(n_1,n_2)$ and $x(n_1,n_2)$, however limited to the values where k_1,k_2 are non-zero. For a 3×3 convolution kernel for example, the cross correlation matrix will be composed of 9 elements.

The right hand side of equation (4.19) represents for the convolution between the Wiener coefficients and the autocorrelation of $x(n_1,n_2)$, also limited to the values where k_1,k_2 are non-zero.

To avoid the convolution operation and to make the equation invertable the autocorrelation matrix is constructed out of the autocorrelation expression. The matrix expression of (4.19) becomes

$$r_{sx}(n_1,n_2) = R_{xx}(n_1,n_2) \,.\, w(n_1,n_2) \tag{4.20}$$

where r_{sx} is the cross-correlation column vector, R_{xx} is the autocorrelation matrix and w stands for the column representation of the Wiener coefficients. Equation (4.20) is the 2D Wiener-Hopf equation.

In 1D this could be written as

$$r_{sx}(n) = R_{xx}(n) \,.\, w(n) \tag{4.21a}$$

where w(n), or $w(n_1,n_2)$ can very easily be found by inverting $R_{xx}(n)$ or $R_{xx}(n_1,n_2)$ and multiplying the result with r_{sx}, i.e.

$$w(n) = [R_{xx}(n)]^{-1} . r_{sx}(n) \tag{4.21b}$$

4.5.3. Wiener Filtering in the Frequency Domain

Instead of constructing the autocorrelation matrix in (4.20), the autocorrelation function of x r_{xx} could be taken. The expression then becomes

$$r_{sx}(n_1,n_2) = r_{xx}(n_1,n_2) * w(n_1,n_2) \tag{4.22}$$

Calculating the Fourier transform of this expression and solving it for $Hw(\Omega_1,\Omega_2)$ which is the spectrum of $w(n_1,n_2)$ results in

$$Hw(\Omega_1,\Omega_2) = P_{sx}(\Omega_1,\Omega_2) / P_{xx}(\Omega_1,\Omega_2) \tag{4.23}$$

Suppose that $s(n_1,n_2)$ is uncorrelated with noise(n_1,n_2) and that $s(n_1,n_2)$ is a zero-mean process. We then obtain :

$$E[s(n_1,n_2)\ ,noise(n_1 - k_1,n_2 - k_2)] = E[s(n_1,n_2)]\ E[noise(n_1 - k_1,n_2 - k_2)] = 0 \tag{4.24}$$

Reconsidering Eq.(4.9) and **omitting the blurring effect of $h(n_1,n_2)$** then taking the correlation with $s(n_1,n_2)$ by using the expectation operator and substituting Eq.(4.24) into it we obtain :

$$E[s(n_1,n_2)\ ,x(n_1 - k_1,n_2 - k_2)\ ] = E[s(n_1,n_2),s(n_1 - k_1,n_2 - k_2)] \tag{4.25}$$

or
$$r_{sx}(n_1,n_2)\ = r_{ss}(n_1,n_2) \tag{4.26}$$

With $s(n_1,n_2)$ and $x(n_1,n_2)$ being uncorrelated, we have

$$r_{xx}(n_1,n_2)\ = r_{ss}(n_1,n_2)\ +\ r_{noise}(n_1,n_2) \tag{4.27}$$

Computing the Fourier transform of Eq.(4.26) and Eq.(4.27) and then substituting the result into Eq.(4.23) produces the well known Wiener filter expression :

$$Hw(\Omega_1,\Omega_2) = P_{ss}\ (\Omega_1,\Omega_2)\ /\ [P_{ss}(\Omega_1,\Omega_2) + P_{noise}(\Omega_1,\Omega_2)] \tag{4.28}$$

When the samples of the process are non zero-mean, the mean value should first be subtracted before filtering and added afterwards [59] :

$$Hw(\Omega) = P_{ss}\ (\Omega)\ /\ [P_{ss}(\Omega) + P_{noise}(\Omega)] \tag{4.29}$$

4.5.4. Wiener filtering in wavelet space

To make the review of the Wiener filtering concept complete it should also be investigated in wavelet space. The wavelet theory is extensively developed in Chapter 5. **For a complete understanding of what follows Chapter 5 should first be read!**

One of the many aspects of wavelets is the fact that after applying a discrete wavelet transform a signal is subdivided into different levels which contains octave spaced, band pass filtered and decimated parts of the signal. Wavelet filtering is in fact a convolution operation with scaled versions of the same impulse response or psf.

For use in Wiener filtering one should start with the approximation that the transfer function could be described by a constant at the different levels and equal to the variance at that level. Considering oversampling, the band between fs/4 and fs/2 contains almost exclusively noise and its variance is used as a measure for the noise in the different bands or levels. In these circumstances Eq. (4.28) becomes

$$Hw\_wav = \sigma^2_{signal\_level} / (\sigma^2_{signal\_level} + \sigma^2_{noise\_level}) \qquad (4.30)$$

4.5.5. Comparative 1D Study

We consider successively

- Wiener filtering in time domain using a limited convolution kernel and application of Eq. (4,21b).
- Wiener filtering as in the first part but with only a Burg approximation of the original signal.
- Wiener filtering in the frequency domain.
- Wiener filtering in wavelet space.

Wiener Filtering in Time Domain

First we consider a 1-D signal s(n) constructed from a Gaussian noise source noise1(n) passed through a lowpass filter . The blurring effect is in this case omitted and only a white noise source noise2(n) with a variance of .084 is added. N is the number of samples in the signal.

$N\_segment \equiv 512$ $n \equiv 0..N\_segment - 1$ $n1 := 0..N\_segment - 2$

$noise1 \equiv gaussn(N\_segment) + .5$

$noise2 \equiv whiten(N\_segment) + .5$
$noise2\_var = var(noise2)$
$noise2\_var = 0.084$

The filtering is realised with a first order single pole IIR construction specified in A1 :

$$\alpha \equiv .9$$

$$A1\_0 \equiv \frac{1}{1-\alpha}$$: DC - amplification

$$A1 \equiv \begin{pmatrix} 1 & 1 \\ 0 & -\alpha \end{pmatrix}$$

$$A1(z) := \frac{z}{z-\alpha}$$: The IIR filter structure

The source signal s(n) results from the response of noise1 from the filter A1. N_segment is the number of samples.

$$s \equiv \frac{response(noise1, A1, N\_segment)}{A1\_0}$$

The deteriorated signal x(n) is created by adding 50% noise. The processes are assumed to be zero-mean. If this is not assumed, the mean value has first to be subtracted from the process and after the filtering added again. The zero-mean process is called xz1 and is created by adding trailing zeros to x1 until the length of the autocorrelation of x1 is obtained.

$$x \equiv s + 1.5 \cdot noise2$$

$x\_mean := mean(x)$ $\quad x1 := x - x\_mean$

$X1 := cfft(x1)$

$s\_mean := mean(s)$ $\quad s1 := s - s\_mean$

$z1_{ni} := 0$ $\quad xz1 := stack(x1, z1)$ $\quad Xz1 := cfft(xz1)$

To estimate the noise in the signal, oversampling is presumed and so the signal between fs/4 and fs/2 contains almost only noise. By separating it in a submatrix and calculating the variance of the ifft :

$$X2 := submatrix\left(X1, \frac{N\_segment}{4}, \frac{N\_segment}{2} - 1, 0, 0\right)$$

$x\_level_2 := icfft(X2)$ $\quad x\_lev1\_var := var\left(x\_level_2\right)$ $\quad x\_lev1\_var = 0.194$

Wiener filtering in the frequency domain can very easily be done by taking the ratio of the cross-power of s, x and the auto-power of x as the transfer function of the filter, provided of course that the original signal s is known.

$xx1 := correl(x1, x1)$ $\quad Pxx1 := cfft(xx1)$

$sx1 := correl(s1, x1)$ $\quad Psx1 := cfft(sx1)$

$$H1 := \frac{\overrightarrow{Psx1}}{Pxx1} \quad : \quad \text{Vector division}$$

$s\_freq := icfft\left(\overrightarrow{(H1 \cdot Xz1)}\right)$ $\quad s\_freq1_n := s\_freq_n + x\_mean$

This technique requires of course a large amount of CPU time due to the FFT's, the point by point division and the IFFT. Time domain filtering will be less time consuming certainly when the value of the filter coefficients tends quickly to zero. Wiener filtering becomes a convolution operation with a limited number of non-zero filter coefficients. Of course the result will not be as good as with the frequency domain filtering .We consider a Wiener filter of length 8. The Wiener filter response results in s_wien.

$N\_wien := 8$ $count\_w := 0..N\_wien - 1$ $i := 0..N\_wien - 1$ $j := 0..N\_wien - 1$

$$Rx1_{i,j} := if\left(i=0, xx1_j, if\left(i \geq j, xx1_{i-j}, xx1_{j-i}\right)\right)$$

$$Rx1_{inv} := Rx1^{-1}$$

$$rsx1_{count\_w} := sx1_{count\_w}$$

$$w1 := 2 \cdot Rx1_{inv} \cdot rsx1 \qquad sum\_w1 := \sum w1 \qquad w1\_norm := \frac{w1}{sum\_w1}$$

$$s\_wien1 := response(x1, w1\_norm, N\_segment) + x\_mean$$

The Wiener filter in Eq.(4.29) was derived on the assumption that the power spectrum of the original signal and noise could be calculated independently. In fact this means that the original signal is known. This is a trivial situation, therefore a technique must be chosen to estimate or approximate the original signal in the best possible way.

Wiener Filtering using Burg's Approximation

Burg's approximation seems to be very appropriate for a 1D problem .A matrix B is constructed containing as denominator the auto-regressive coefficients of the Burg approach and a constant as numerator . The noise is estimated in the (fs/4, fs/2) band :

$N\_burg := 9$ $B_{den} := burg(x1, N\_burg)$

$count\_b := 0..N\_burg$

$$B_{num_{count\_b}} := if(count\_b = 0, 1, 0) \qquad B := augment(B_{num}, B_{den})$$

The frequency components of B are calculated by first producing its impulse response of B and then calculating its FFT :

$$delta_n := if(n = 0, 1, 0) \qquad b := response(delta, B, N\_segment)$$

$$B_{freq} := cfft(b)$$

H2 is the frequency domain expression for the Wiener filter. To go back to the time domain and to do the convolution with a limited number of coefficients ,decimation of the frequency components should take place before applying IFFT.

Trying to work with a substitute of the original signal s(n) by sending noise through the Burg filter and then working with the cross correlation of this substitute of s(n) and x(n) does not give good results. One can of course create a signal with approximately the same frequency content as s(n) but, due to the Fourier transform constraints, one cannot put the harmonics at the right place in the time domain by applying an IFFT .

$$P\_burg := \overrightarrow{(B_{freq} \cdot \overline{B_{freq}})}$$: Vectorial product of B_{freq} and its complex conjugate.

$$H2 := \frac{\overrightarrow{P\_burg}}{(P\_burg + x\_lev1\_var)}$$

$$N\_decim := \frac{N\_segment}{N\_wien}$$: Decimation factor

$$H2_{decim} := resample(Re(H2), 2 \cdot N\_decim, 2)$$: Subsampling

$$w2 := \text{icfft}(H2_{decim})$$

$$sum\_w2 := \sum w2 \qquad w2\_norm := \frac{w2}{sum\_w2}$$

$$s\_wien2 := response(x1, w2\_norm, N\_segment) + x\_mean$$

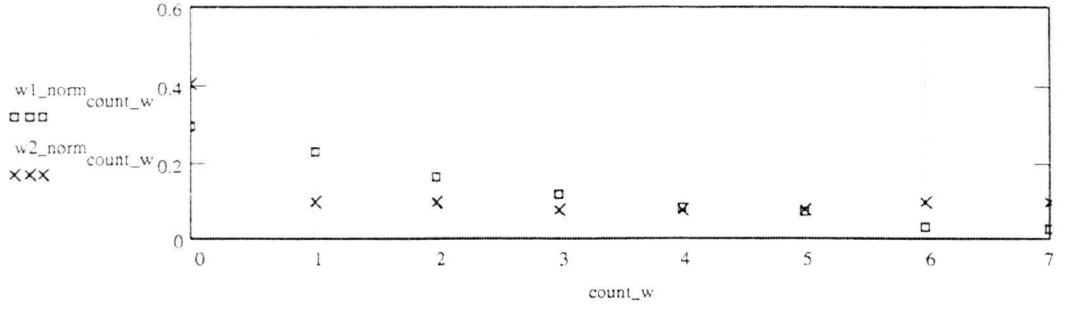


Fig. 4.13 Comparison between w1 : Wiener coefficients calculated from known signal situation and w2 : Wiener coefficients using Burg's approximation of the signal. The difference in Wiener coefficients reveals the rather poor estimation by Burg's approximation of the spectrum of s(n).

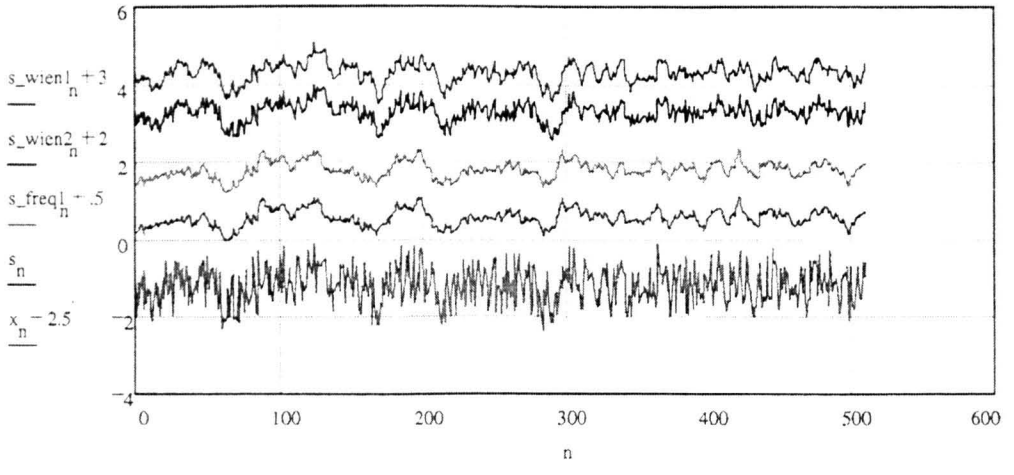


Fig. 4.14 Comparison between different Wiener filtering results. s_n is the pure signal, x_n stands for the signal + noise. s_wien1 is the result of Wiener filtering in the time domain using a limited kernel of 8 coefficients and, knowing the original signal. s_wien2 does the Wiener filtering in the time domain using only a Burg approximation of the original signal. s_freq1 gives the best result with the frequency domain filtering.

Relying on a subjective comparison between the original, degraded and processed signals can sometimes lead to good choices for the right filter but it can be cumbersome to express these ideas in numbers. One of the objective criteria for comparison is the **normalised mean square error (NMSE)** leading to an estimation of the **signal to noise ratio improvement (SNRimpr)** . Another one is the **correlation coefficient**. All described in [59]. Note that these are just a few of the many possible objective measures and they can be misleading.

$$NMSEsx := 100\cdot\frac{var(s - x)}{var(s)}$$

$$NMSEs\_wien2 := 100\cdot\frac{var(s - s\_wien2)}{var(s)}$$

$$NMSEs\_wien1 := 100\cdot\frac{var(s - s\_wien1)}{var(s)}$$

$$NMSEs\_freq1 := 100\cdot\frac{var(s - s\_freq1)}{var(s)}$$

$$SNRimpr1 := 10\cdot\log\left(\frac{NMSEsx}{NMSEs\_wien1}\right)$$

$SNRimpr1 = 6.363$

$$SNRimpr2 := 10\cdot\log\left(\frac{NMSEsx}{NMSEs\_wien2}\right)$$

$SNRimpr2 = 5.914$

$$SNRimpr3 := 10\cdot\log\left(\frac{NMSEsx}{NMSEs\_freq1}\right)$$

$SNRimpr3 = 284.81$

$cor\_coef1 := corr(s, s\_wien1)$

$cor\_coef1 = 0.611$

$cor\_coef2 := corr(s, s\_wien2)$

$cor\_coef2 = 0.571$

$cor\_coef3 := corr(s, s\_freq1)$

$cor\_coef3 = 1$

The signal to noise ratio improvements (6,363←→5.914) and correlation coefficient (0.611←→0.571) give comparative results. Compared with the visual results in Fig.4.14 an improvement of the signal to noise ratio is indeed realised. The

shape of the signal has changed and this is reflected in the rather poor correlation coefficients.

Wiener Filtering in Wavelet Space

Wavelet filtering is based on the fact that a wavelet transform results in a lossless splitting up of the signal in octave bands also called levels. [57] When N_segment is the number of samples then every level contains $N\_level/2^{level\_order}$ samples . When the number of samples is for example 512 then level 1 extends from fs/4 until fs/2 and contains 256 samples. Level 2 extends from fs/8 until fs/4 and contains 128 samples, etc. For this application the Daubechies4 is chosen. An expansion is made by repeating every value a number of times so that all expanded levels contain the same number of samples and that they can be compared.

$$X\_wavelet \equiv dwavelet(x)$$

$$wavelet\_levels \equiv \frac{\log(N\_segment)}{\log(2)} \qquad w\_l \equiv wavelet\_levels - 2 \qquad n4 \equiv 0..w\_l$$

$$X\_wav\_expand_{n4,n} \equiv X\_wavelet_{2^{n4+1}+floor\left(\frac{n}{2^{w\_l-n4+1}}\right)}$$

Note that the amplitude differences stand for more or less important coefficients in the wavelet space. In fact they mean that in some frequency bands the amplitudes are higher than in other ones.

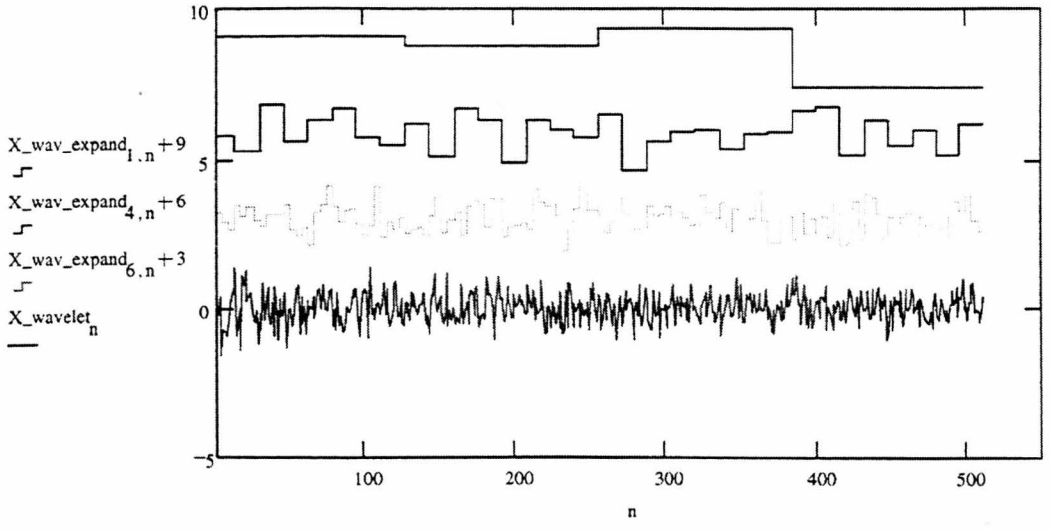


Fig.4.15 Wavelet expansion of signal x at level 7 with 4 samples, level 4 with 32 samples and level 2 with 128 samples

First we calculate the noise power (variance) of the signal in the highest frequency band (level 1 or also fs/4-fs/2)

$$X\_var\_noise \equiv var\left[\left(X\_wav\_expand^{T}\right)^{\langle w\_1\rangle}\right] \qquad X\_var\_noise = 0.183$$

$$n4 \equiv w\_1..\,1$$

We now apply Eq. 4.30 at all levels

$$H\_wav\_wien_{n4-1} \equiv \left|\begin{array}{l} X\_var\_signal\_noise \leftarrow var\left[\left(X\_wav\_expand^{T}\right)^{\langle n4\rangle}\right] \\ X\_var\_signal \leftarrow X\_var\_signal\_noise - X\_var\_noise \\ Wien\_filt \leftarrow \dfrac{X\_var\_signal}{X\_var\_signal\_noise} \\ 0 \text{ if } Wien\_filt \le 0 \\ Wien\_filt \text{ otherwise} \end{array}\right.$$

The wavelet attenuation coefficients which have to be applied at the different levels are collected in the vector H_wav_wien. Starting with level 7 ; it contains 4 elements and represents the band (fs/256 - fs/128). It is attenuated to 67.4% of its original level. Level 4 contains 32 elements, represents the band (fs/32 - fs/16) and is attenuated to 43.3% of its original level.

$$H\_wav\_wien = \begin{bmatrix} 0.674 \\ 0.491 \\ 0.723 \\ 0.433 \\ 0.226 \\ 0.183 \\ 0 \end{bmatrix}$$

X_inv_filt_wavelet := idwavelet(X_filt_wavelet)

X_filt_wavelet realises the multiplication with the attenuation coefficients at all levels. Idwavelet performs the inverse wavelet transform.

```
X_filt_wavelet := for n5 ∈ w_l - 1..3
                    for k ∈ 2^n5 .. 2^(n5+1) - 1
                      X_wav_k ← X_wavelet_k · H_wav_wien_(n5-2)
                  for k ∈ 0..w_l - 2
                    X_wav_k ← X_wavelet_k
                  X_wav
```

Some 'objective' measurements :

$$NMSEs\_wavelet := 100 \frac{var(s - X\_inv\_filt\_wavelet)}{var(s)}$$

$$SNRimpr3 := 10 \cdot \log\left(\frac{NMSEsx}{NMSEs\_wavelet}\right)$$

SNRimpr3= 8.154

cor_coef4 := corr(s, X_inv_filt_wavelet)

cor_coef4 = 0.602

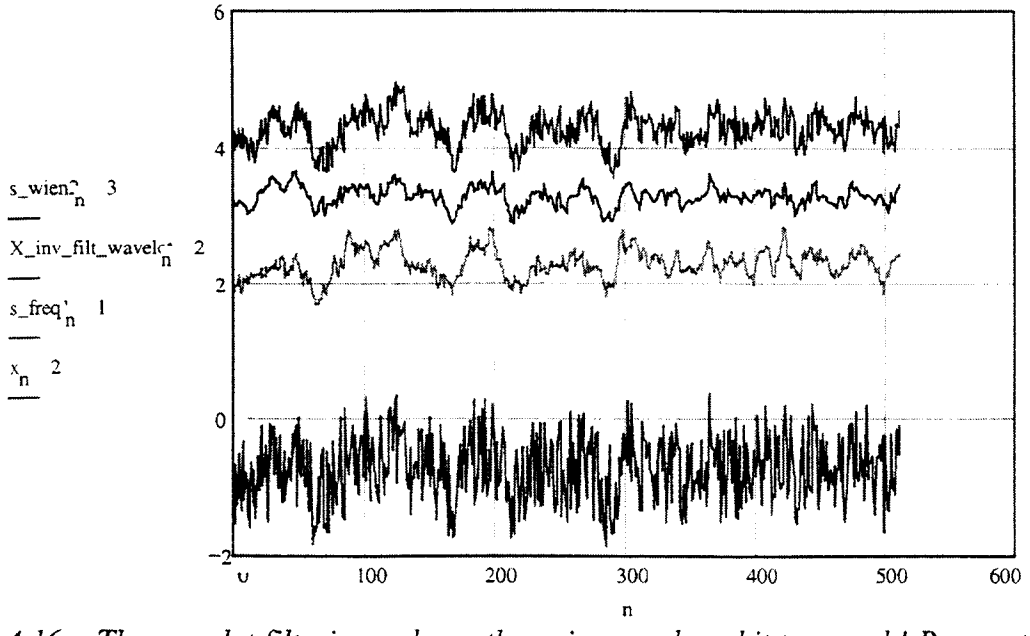


Fig.4.16. The wavelet filtering reduces the noise, maybe a bit too much! Parameters should be included in the classical Wiener expression to diminish the effect.

Some conclusions : Wiener filtering in wavelet space looks promising . The signal to noise ratio improvement is better than with the classical Wiener filter coefficients. Lim [59] suggests to adapt the Wiener filter formula (4.29) to :

$$Hw(\Omega_1,\Omega_2) = \{P_{ss}(\Omega_1,\Omega_2) / (P_{ss}(\Omega_1,\Omega_2) + \alpha.P_{noise}(\Omega_1,\Omega_2)\}^{\beta}$$

with α and β case empirical parameters to be determined case by case. This will be implemented into further research. We will not go further into more details on this matter, instead we will investigate a 2D case.

4.5.6. 2D Wiener filtering example

In this study we compare the results of space domain Wiener filtering with a limited 9×9 kernel $w(n_1,n_2)$ and the Wiener filtering in wavelet space. The autocorrelation matrix R_{xx} and the cross correlation matrix r_{sx} are constructed in a Matlab program called WIEN2D2. This program was translated and incorporated into a more general object oriented C++ program called 2D DSP [45]. W is the point spread function.

w =

0.4298	0.1722	0.0739	0.0426	0.0348	0.0364	0.0349	0.0328	0.0270
0.2076	0.0492	-0.0096	-0.0214	-0.0202	-0.0168	-0.0169	-0.0237	-0.0253
0.0930	0.0066	-0.0072	-0.0136	-0.0105	-0.0081	-0.0119	-0.0131	-0.0189
0.0519	-0.0051	-0.0104	-0.0043	-0.0030	-0.0049	-0.0047	-0.0060	-0.0104
0.0382	-0.0121	-0.0054	-0.0003	-0.0024	-0.0039	-0.0065	-0.0014	-0.0056
0.0236	-0.0130	-0.0025	-0.0013	-0.0006	-0.0026	0.0006	-0.0022	-0.0006
0.0198	-0.0163	-0.0020	0.0013	-0.0012	0.0009	0.0003	0.0003	0.0008
0.0154	-0.0167	-0.0067	-0.0012	0.0022	0.0000	0.0007	0.0010	-0.0016
0.0087	-0.0257	-0.0066	-0.0041	-0.0006	-0.0010	0.0027	0.0027	-0.0018

Fig.4.17. original Lena picture

The Lena picture (fig.4.17) is deteriorated by 30% noise in Fig.4.18.

Fig.4.18. Lena deteriorated with 30% noise.

The amplitude of the spectrum of w is presented in Fig.4.19. A fftshift operation is performed on the figure so that the DC level occurs in the centre of the frequency plane.

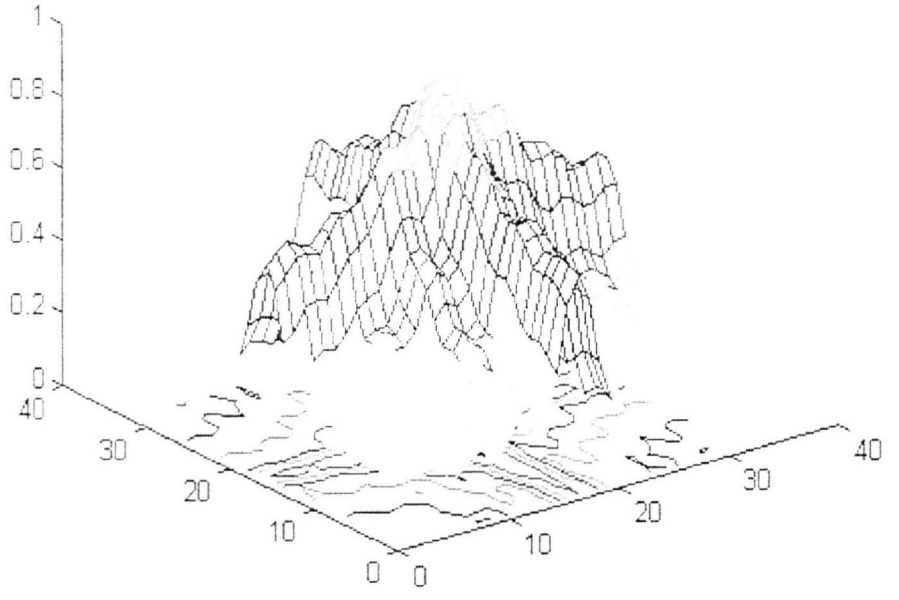


Fig.4.19. Amplitude spectrum of Wiener coefficients $w(n_1,n_2)$.

The result of the filtering is shown in Fig.4.20. This result is rather disappointing and it does not improve very much by making the kernel larger. A 3×3 kernel gives almost the same results. This is to be expected since the noise is spread all over the spectrum and when it is of the same frequency as the original signal it becomes very difficult to remove .

Fig.4.20. Lena, Wiener filtered with a 9×9 convolution kernel

Wiener filtering in wavelet space gives a better result. Fig. 4.21. is the wavelet transform of Lena. For simplicity, the Haar transform was performed until level 3. Special Mathcad functions were written to do the wavelet transform based on Mallat's scheme. The functions are called wavelet_anal and wavelet_synt and have as arguments the picture name, the wavelet and the level (see Chapter 6) This scheme, but further extended into wavelet packages was also applied in a C program on a TMSC80 processor (see Chapter 7), the intention being to make a real time application.

picture := READBMP(lena256)

pict_wav := wavelet_anal(picture, Haar, 3) : 2D Haar wavelet transform on picture until level 3.

The 2D wavelet transform with Mathcad tools is described in detail in chapter 6.

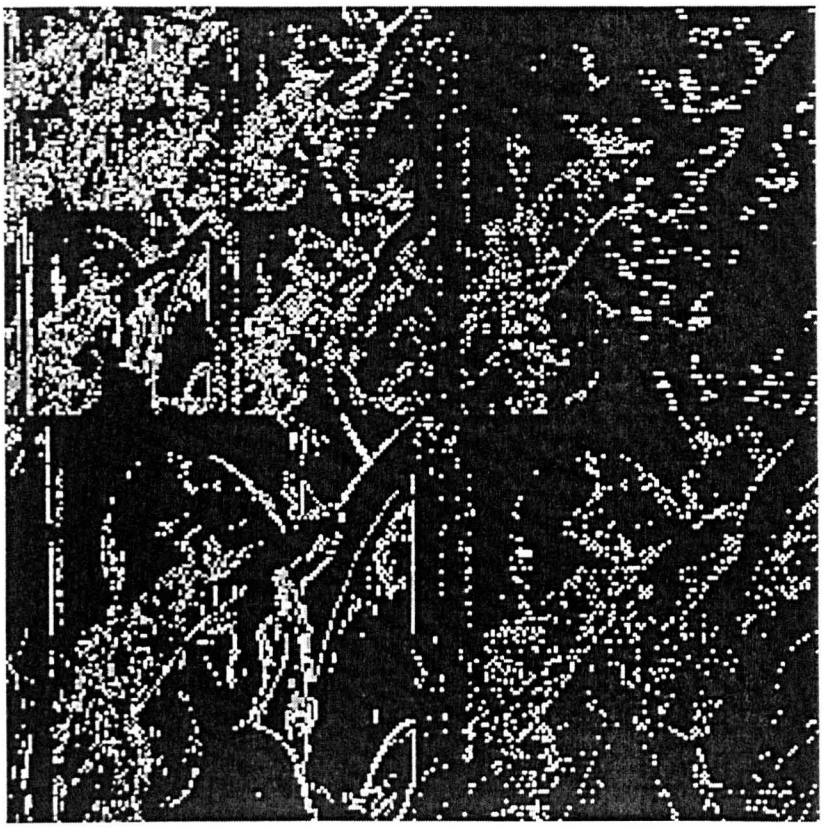


Fig.4.21. Wavelet transformed Lena picture into level 2. Lena contains 256×256 pixels. Lena is filtered and down-sampled (:2) (keep only the even samples) twice to produce the levels 1 and 2. The square picture in the right bottom (128×128) contains the high pass filtered and down-sampled part of Lena along both axis $[n_1,n_2]$ *:* $(HH)_{level\_1}$*. The left bottom picture (128×128) contains the high pass filtered and down-sampled part along the* n_1 *axis and the low pass filtered and down-sampled part along the* n_2 *axis of Lena* $(HL)_{level\_1}$*. The right upper corner picture presents the opposite filtering of the former* $(LH)_{level\_1}$*. The left upper corner picture should contain the low pass and down-sampled version of Lena* $(LL)_{level\_1}$*, but it is by itself subdivided into 4 new equal square pictures containing once again* $(HH)_{level\_2}$*,* $(HL)_{level\_2}$*,* $(LH)_{level\_2}$ *and* $(LL)_{level\_2}$*. This last one is , in the case of 3 levels is subdivided in* $(HH)_{level\_3}$*,* $(HL)_{level\_3}$*,* $(LH)_{level\_3}$ *and* $(LL)_{level\_3}$*. (Note : In DSP literature the down-sampling operation is very often referred to as a decimation operation. The converse operation is referred to as up-sampling or also interpolation .)*

.

30% noise is added to the picture to produce a noisy_picture and then a wavelet transform is once again performed.

$dim := cols(picture)$

$i := 0..dim - 1$

$j := 0..dim - 1$ $\quad noise_{i,j} := 2.56 \cdot rnd(1)$

$k := 0..\frac{dim}{2} - 1$

$noise\_percent := 30$

$noisy\_picture := picture + noise\_percent \cdot noise$

$noisy\_pict\_wav := wavelet\_anal(noisy\_picture, Haar, 3)$

The variance of noise and signal + noise at all levels are calculated in a similar way as in the 1D case.

$lev\_1\_var = 824.637$

$lev\_2\_var = 304.404$ $\quad att\_lev\_2 := \frac{lev\_2\_var - \frac{lev\_1\_var}{4}}{lev\_2\_var}$

$lev\_3\_var = 301.051$ $\quad att\_lev\_3 := \frac{lev\_3\_var - \frac{lev\_1\_var}{16}}{lev\_3\_var}$

$att\_lev\_1 := .2$ $\quad att\_lev\_2 = 0.323$ $\quad att\_lev\_3 = 0.829$

Although the attenuation level at level 1 should be put to 0 (i.e. noise between $f_s/4$ - $f_s/2$) a more empirical approach is to keep some information of level 1 in the picture to fill in the details without too much emphasis on the noise. The attenuation levels suggest a low pass action : 82.9% of amplitude of wavelet coefficients in band $f_s/16$-$f_s/8$; 32.3% of amplitude of wavelets in band $f_s/8$-$f_s/4$ and 20% in level 1 band.

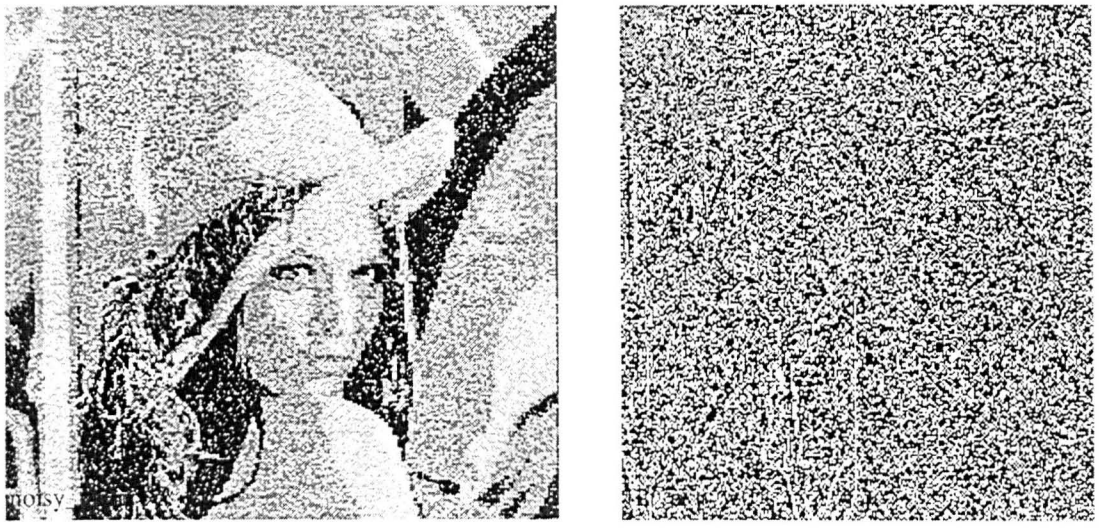


Fig.4.22. Noisy Lena and its 3 level Haar wavelet transformed picture. The different levels are already 'Wiener attenuated'.

This Wiener filtered wavelet transformed picture has to be transformed back with an inverse wavelet transform :

```
wiener_wav_n_p := wavelet_synt ( filt_n_p_w0, Haar_rev , 3 )
```

The result is shown in Fig.4.23 . Fig.4.23a shows the original Lena ; Fig.4.23b is the 'denoised' version of Lena. Note that some detail is lost but that the noise is certainly minimal. The result is much better than with the autocorrelation based Wiener filter. The technique is rather simple and certainly more sophisticated algorithms than simple attenuation's are possible at the different levels. However because of the combination of its relative high quality and simplicity it will be implemented on a video processor system (VSP1 from Philips)

Fig.4.23. Original Lena and its Wiener wavelet filtered picture.

4.6. Conclusions

The results of the combination of Wiener filtering and wavelet analysis are rather promising. Therefore the forthcoming chapter on wavelets describes in detail the different discrete fast wavelet transforms studied. In Chapter 7 they are implemented on different video processors : VSP1 from Philips and TMSC80 from Texas Instruments.

CHAPTER 5

Wavelet Transform

5.1 Introduction

Transform techniques , once presented under different names like time-frequency analysis, multiresolution analysis, subband coding, pyramid algorithms are now unified in one framework called **wavelet analysis**. It is in fact a multi-disciplinary activity which uses mathematics, computer sciences and signal processing. Its goal is to reveal more relevant information than with classical techniques and provide efficient algorithms for non-stationary problems in the field of signal and image processing.

In this chapter we investigate the shortcomings of classical Fourier analysis and present some solutions by using **short-time Fourier transform (STFT)** . The choice of Gauss windows leads to the **Gabor transform**. Then we define to the **wavelet transform (WT)**. Basic difference is the use of a single window for STFT and a multiple window condition (short windows at high frequencies and long windows at low frequencies) for WT analysis. Most useful in numerical analysis is the **discrete** wavelet transform, as it is derived from the **continuous** wavelet transform. The evolution from one to the other is explained. Before looking to orthogonality we consider frames for STFT and WT. Under the wavelets we introduce some simple wavelets like the **Haar wavelet**. **Orthonormal bases of wavelets** and **multiresolution analysis** are also investigated then. **Compactly supported wavelets** are very useful for numerical analysis, **Daubechies wavelets** and others are presented. Finally we investigate the **symmetry problem** and introduce the **biorthogonal wavelets** (more details in Chapter 6). **Fast wavelet transforms (FWT)** eliminate redundancy in the calculations. We follow Mallat's scheme to realise the FWT.

We restrict ourselves to real-valued, measurable and square integrable functions of 1 and 2-D variables.

5.2 Short Time Fourier Transform (STFT)

Signals x(t) whose statistical properties do not evolve in time, also called stationary signals, find a 'natural' transform in the classical Fourier transform.

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt \tag{5.1}$$

As shown in Eq. (5.1) the Fourier coefficients are computed as inner products of the signal with sine/cosine wave basis functions of infinite duration (stationary signals). So it is not surprising that Fourier analysis only works well if x(t) is built up out of stationary components. Any abrupt change in x(t) (non stationary behaviour) is spread out in the analysis all over the frequency axis in X(f). Therefore, an appropriate analysis for non stationary signals is required.

The first idea is to create a **local** Fourier transform by looking at the signal through an in time sliding window. The signal looks as though composed of approximately stationary time segments.

The Fourier transform Eq.(5.1) was first adapted by Gabor to define the STFT(τ,ϕ) as follows . Consider a window g(t) of limited extent and centred at time location τ. The Fourier transform of the windowed signal x(t)g(t-τ) yields the short-time Fourier transform (STFT) :

$$STFT_x(\tau,\phi) = \int_{-\infty}^{\infty} x(t)g(t-\tau)e^{-j2\pi\phi t}dt \tag{5.2}$$

The signal x(t) is mapped into a 2-dimensional function in a time-frequency plane (t,ϕ). The analysis looks very similar to the classical Fourier transform, however the choice of g(t) will influence the analysis considerably .

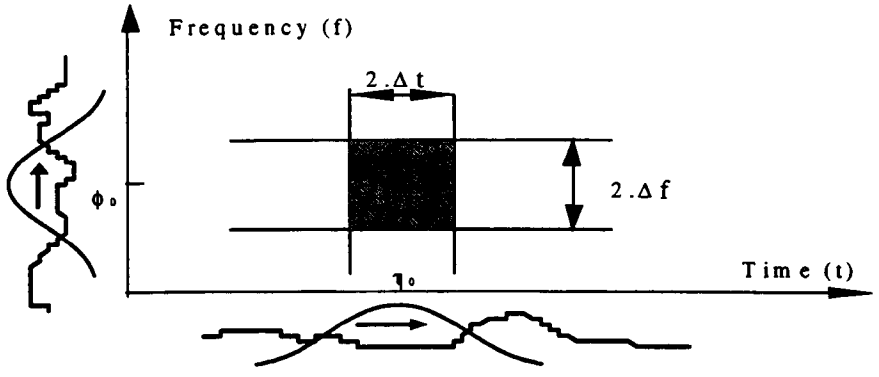


Fig. 5.1. Analysis domains with time and frequency windows sliding over data functions in time and frequency domain. The resolution rectangle has an area of 4.Δf.Δt.

Figure 5.1 shows the time frequency plane with vertically, around τ_0 , a windowed version of x(t) with all its frequency contents calculated at τ_0. Alternatively one could consider $g(t)e^{-n2\pi\phi t}$ as the impulse response of a bandpass filter centred at frequency ϕ_0. The STFT acts in this case as a convolution operation and produces a bandpass filtered version of x(t). Around the frequencies ϕ_0 one can horizontally see the modulated filter banks able to detect the time contents in x(t) at ϕ_0 .

5.2.1 Resolution

An immediate drawback must be made to the time and frequency resolution. How accurately are the time and frequency components known, or what is the ability of the STFT to distinguish 2 close sine waves? For a window function g(t) and its Fourier transform G(f) we define the bandwidth Δf of the window in a rms sense as

$$\Delta f^2 = \frac{\int_{-\infty}^{\infty} f^2 (|G(f)|)^2 df}{\int_{-\infty}^{\infty} (|G(f)|)^2 df} \tag{5.3}$$

Considering Parseval's theorem, the denominator stands for the energy in g(t). So, discrimination between 2 sine waves will only be possible if they are more than Δf apart. In a similar manner, the spread in time noted by Δt can de defined .

$$\Delta t^2 = \frac{\int_{-\infty}^{\infty} t^2 (|g(t)|)^2 dt}{\int_{-\infty}^{\infty} (|g(t)|)^2 dt} \tag{5.4}$$

The denominator stands again for the energy in g(t). Note that the time-window sizes in Eq. (5.3) and (5.4) are radii about their centre. Therefore, when referring to the window width, one traditionally doubles the radius. (See Fig 5.1.).

Resolution in time and frequency cannot be arbitrarily small. Similar to physics there is an uncertainty principle for waves, stating that there is a lower bound on their product.

$$\Delta t \Delta f \geq \frac{1}{4\pi} \tag{5.5}$$

It is also referred to as the **Heisenberg uncertainty principle**. As $\Delta t.\Delta f$ has the dimension of an area and the shape of a rectangle it is called the **Heisenberg box**. Apparently there is a trade off between better time resolution and worse frequency resolution or vice versa. Inappropriate choice of the window can have far-reaching consequences ; some important considerations are :

- The area can be arbitrarily large resulting in unnecessary overlapping and a huge amount of redundancy in the analysis.

- Gaussian windows meet the lower bound of equality and were therefore chosen by Gabor (Eq. (5.7)). The STFT is in this case called the **Gabor Transform.**

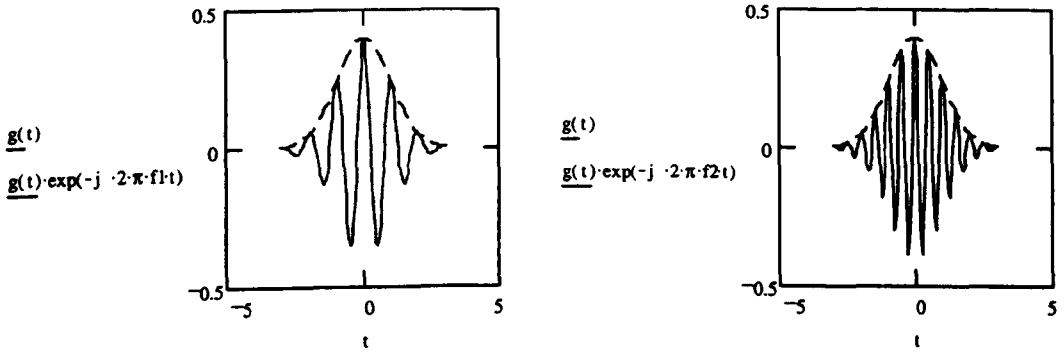
5.2.2 Gabor Transform

Equation (5.6) and Fig. 5.2 show how a Gaussian time-window is transformed in a Gaussian frequency-window. It is calculated using fft on N=121 samples with sampling time t_s=0.05sec. σ is put to 1 and f_1 and f_2 are respectively 1 and 2 Hz. The time-axis is in seconds, the frequency axis in Hz.

$$g(t) = \frac{e^{-\frac{t^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma} \qquad \text{Fourier Transform} \rightarrow \qquad G(f) = e^{-(\sqrt{2}\pi\sigma f)} \qquad (5.6)$$

The resolutions for Gaussian windows are

$$\Delta f = \frac{1}{2\sqrt{2}\pi\sigma} \qquad \Delta t = \frac{\sigma}{\sqrt{2}} \qquad \Delta t \Delta f = \frac{1}{4\pi} \qquad (5.7)$$



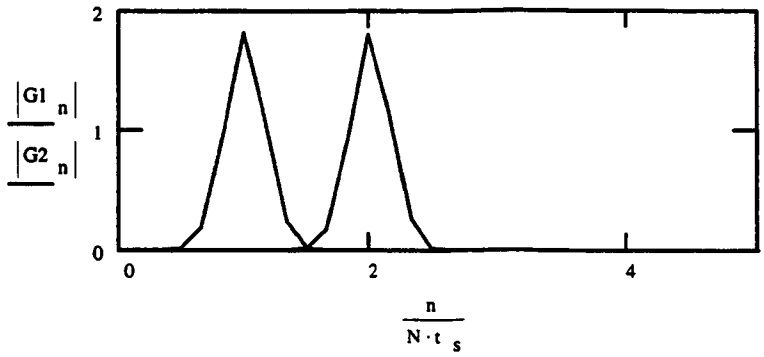
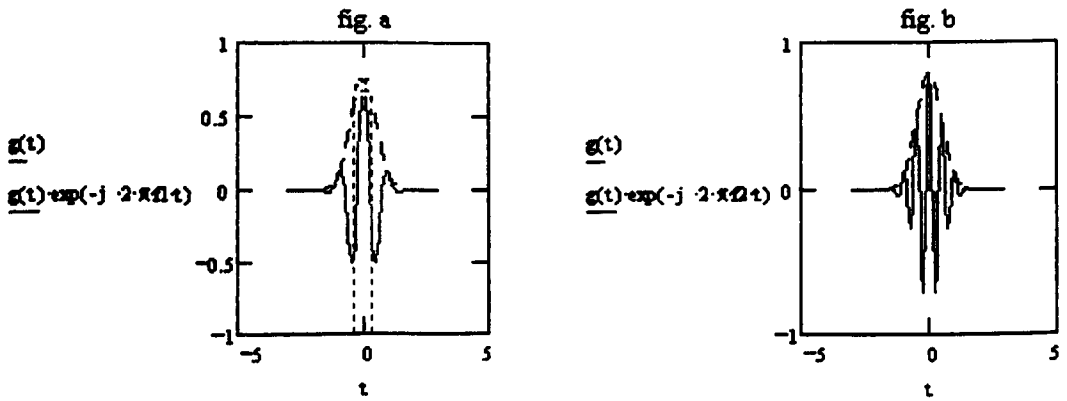


Fig.5.2 Modulated Gaussian windows (fig. a&b) and their spectra (fig. c) ,with : (Δf=0.112Hz,Δt=0.75sec).

Varying σ can change the resolution in time domain , however as σ is present in the denominator of Δf and the numerator of Δt, their product is σ independent and the Heisenberg inequality stays fulfilled. Fig. 5.3 shows the consequences in frequency domain. (σ :1→0.5) Remark that if the 2 peaks in frequency domain would come as close as Δf one couldn't distinguish ϕ_1 from ϕ_3 any more. (Fig. 5.3.d) For the same ϕ_1 and ϕ_2 as in Fig. 5.3 the resolutions now become : 2.Δf=0.450Hz, 2.Δt=0750sec. (between dotted lines in fig. c and fig. a)



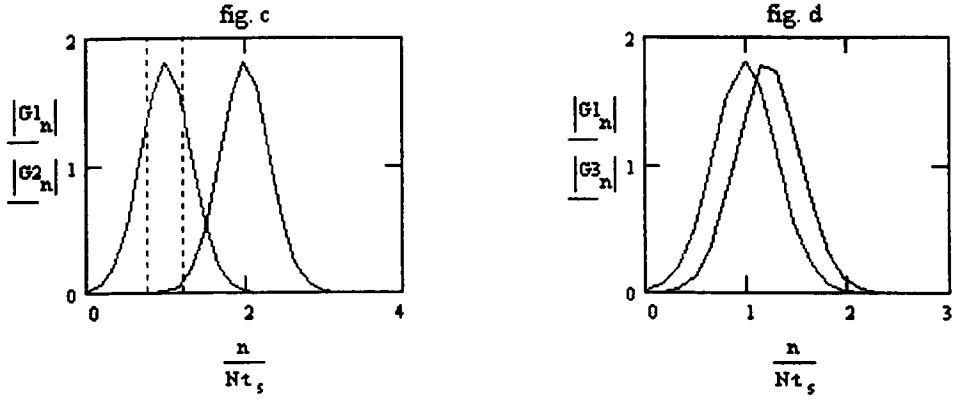


Fig.5.3 Modulated Gaussian windows and their spectra.($\sigma = .5$)

5.3 Frames

When coming to practical realisations also for wavelets, some not yet answered questions, arise :

- In which way does the analysis cover the whole signal? Or, what is the maximum allowable step size (τ_0) compared with the size (read : resolution Δt) of a sliding window to avoid gaps on the time axis?
- With a certain Δt there is a Δf. How to choose the frequency spacing (ϕ_0) relative to Δf to loose no frequency information?

For numerical use the Eq. (5.2), standing for the continuous version of the STFT, should be adapted in a first discretization by only considering the STFT at discrete time ($p.\tau_0$) and frequency points ($q.\phi_0$) equi-spaced in a time-frequency plane. Spacing in time is called τ_0, spacing in frequency is called ϕ_0 . The discretized STFT can therefore be seen as a function of 2 integers variables (p,q) . This is defined in Eq.(5.8).

$$S_x(p,q) = \int_{-\infty}^{\infty} x(t)g(t - p\tau_0)e^{-j2\pi q\phi_0 t}dt \qquad (5.8)$$

The discrete STFT can be seen as convolutions of the time signal x(t) with members of a family of functions $g_{pq}(t)$.

$$g_{pq}(t) = g(t - p\tau_0)e^{-j2\pi q\phi_0 t} \tag{5.9}$$

More compactly Eq. (5.8) becomes

$$S_x(p,q) = \int_{-\infty}^{\infty} x(t)g_{pq}(t)dt \tag{5.10}$$

Note that we are still working with continuous functions and integrals. A second step will be take place when we substitute $n.t_s \rightarrow t$. (t_s : sampling time).

Particularly useful in this circumstance is the use of scalar product notation .

$$S_X(p,q) = \langle x, g_{pq} \rangle \tag{5.11}$$

This notation will be very useful when we will formulate severe constraints on $g_{pq}(t)$ so it could become a basis for decomposing and reconstructing x(t). Mathematically this is formulated as : For $x(t) \in L^2(\Re)$ (square integrable also called Lebesgue functions) we would like the set $g_{qp}(t)$ to constitute a basis in $L^2(\Re)$ for decomposition by using $S_x(p,q)$. We are heading for it, but for the moment we are a little less demanding and instead investigate **non-independent** vectors , also called **frames**, which allows **lossless** analysis when calculating the STFT at the discrete lattice points ($p\tau_0, q\phi_0$).

Let $\{\psi_i\}$ be a possible infinite collection of elements of a Hilbert space **H**. If $\{\psi_i\}$ would constitute an orthonormal basis in **H** then :

$$\| x \|^2 = \sum |\langle x, \psi_i \rangle|^2 \qquad (5.12)$$

$\| x \|^2$, called the squared length of x, but also the energy in that signal .

If $\{\psi_j\}$ doesn't establish an orthogonal basis maybe it can still act as a set of non-independent vectors? The equation (5.12) can then be a little relaxed and the expression can be rewritten as :

$$A . \| x \|^2 \leq \sum |\langle x, \psi_i \rangle|^2 \leq B . \| x \|^2 \qquad (5.13)$$

A and B are called the positive frame bounds, if they are finite , $\{\psi_i\}$ constitutes a frame. The frame bounds can be calculated [57] and conclusions can be drawn from the ratio B/A about the quality of reconstructability of the signal x(t) :

- B>A : This ratio is a degree of instability. Efforts should be made to minimise the ratio. Reconstruction is possible but can numerically become largely unstable. For details see [57].
- B=A : This is called a **tight frame** situation. A is a measure of the redundancy of the frame.
- B=A=1 : Redundancy is minimal and ψ_i forms an **orthonormal basis.**

Next to these frame bounds there are of course the lattice spacings defined by τ_0 and ϕ_0. Common sense tells us that there should be overall density of the lattice to cover the whole signal. However, it is not so easy to formulate stringent conditions. The following conditions given now are **necessary** , but **not sufficient** for all window functions. [60]

5.3.1 Practical considerations

- It can be shown [60] that if $\tau_0 . \phi_0 < 1$ then there is a chance to have a frame. This condition means that the period of ϕ_0 must be larger than the time spacing τ_0.

- The covering should be complete which means that if τ_0 is to large compared with the window width of g(t) there can be gaps on the time axis. Overlapping is necessary. (1.5 times seems to be a good overlapping coefficient). With the Gabor transform , the variance σ plays an important role. σ must be large enough (see Eq. (5.7)) to make Δt sufficiently overlapping. This leads inevitable to smaller Δf with possible gaps in the frequency domain . Intuitively this leads us to considerations concerning $\Delta t/\tau_0$ and $\Delta f/\phi_0$. Making them as equal as possible to each other will give the best results. This means that **the shape of the resolution rectangles should be similar to the layout of the lattice points.**
- **Optimal shape** : make $\tau_0/\phi_0 = \Delta t/\Delta f$ and use the resolution identities for Gabor transforms ($\Delta t = \sigma/\sqrt{2}$, $\Delta f = 1/(2\sqrt{2}.\pi\sigma)$) . The optimal variance is :

$$\sigma = \frac{\sqrt{\tau_0}}{2\pi\phi_0} \tag{5.14}$$

- The window function g(t) must decay quickly as $t \rightarrow \infty$, at least as fast as $(1+|t|)^{-3}$
- In discrete analysis Shannon's sampling theorem should of course be respected as well.

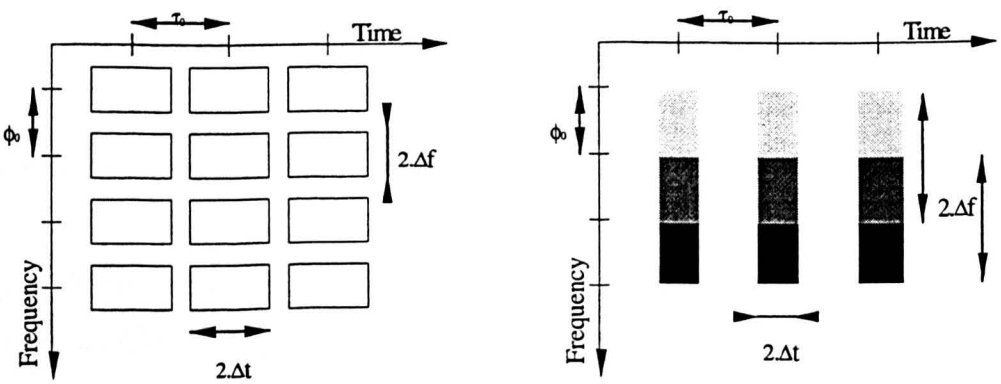


Fig 5.4. Relative good proportional spacing. A slight overlap would be better. (Left figure) The right figure illustrates a bad choice of $\Delta t/\Delta f$ resulting in gaps in time domain and redundancy (overlapping) in frequency domain.

It is not so surprising that it results in coercion's on τ_0 and ϕ_0 . The more closer we space the lattice, the more information we capture.

5.4 Continuous Wavelet Transform (CWT)

5.4.1 From STFT to CWT : The Morlet Wavelet

To get around the resolution limitation while still respecting Heisenberg's inequality, one could imagine the gradual and proportional change to Δf and Δt while completing the analysis.

This would lead us to a **multiresolution analysis** . How can this be realised? We saw that by varying σ in the Gaussian window the resolutions changed . Looking now at nature and wondering how for instance sound waves appear, one perceives very often 'long' low-frequency waves and 'short' high frequency waves. So, why not modify the Gabor transform into a multiresolutional analysis and put the frequency f into the quadrature exponential expression! This concept will lead us first to the **Morlet wavelet** [19] and later, after a generalisation of the concept of multiresolutional analysis to **wavelets** with **orthogonal basis**.

$$g(t,f)e^{-j2\pi ft} = \frac{e^{-(\frac{tf}{\sqrt{2}\sigma})^2} e^{-j2\pi ft}}{\sqrt{2\pi}\sigma} \tag{5.15}$$

For f_1, f_2, f_4 respectively being 1, 2, 4 Hz and $\sigma = 1$ the multiresolutional windows look like in Fig. 5.5. Remark the equal number of oscillations in the 3 cases. They are in fact **scaled versions** of each other.

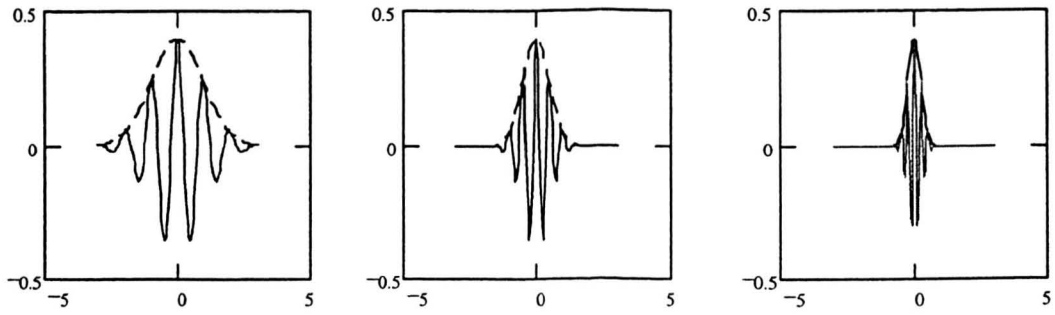


Fig. 5.5 Modulated multiresolutional windows : f = 1, 2, 4 Hz in Eq. (5.15).

The associated filter bank representations , with scale modification, are drawn in Fig. 5.6.

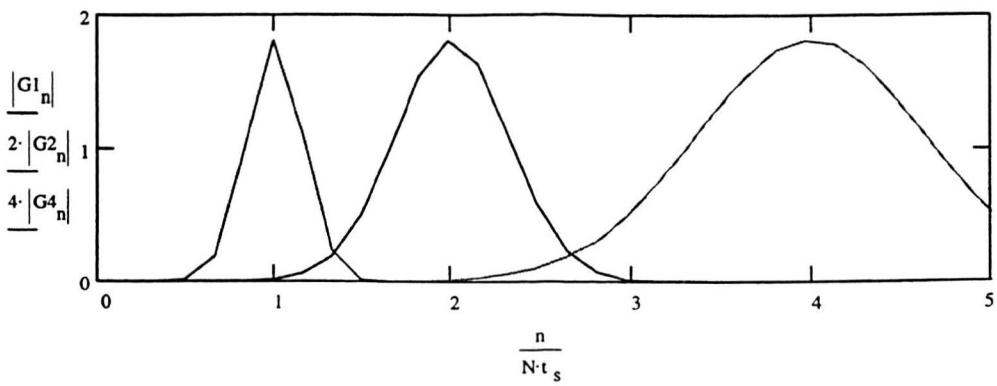


Fig. 5.6. Spectrum division of wavelet-based bandpass filtering. (Δf/f =constant)

5.4.2 Morlet Wavelet

The Morlet wavelet is really a step between the STFT and the WT (wavelet transform). The former uses local complex exponential functions to modulate the analysing function and to create an inner product with it. The latter instead uses very much impelled functions like for instance the block function (Haar) or spline functions to be translated, dilated to finally make the inner product with the analysing function. Fig. 5.7. shows the real and imaginary part of the Morlet wavelet at different scales. Remark that the Morlet as the Meyer one are in fact infinite length wavelets, compared with the Daubechies, Coifman, .. which are of a compact support form. The wavelet, constructed here, is build with a similar expression as Eq. (5.15). The basis

frequency at scale 1 is 1 Hz. The variance σ is chosen in such a way that the Gaussian window almost completely attenuates the oscillatory signal after 6 time units (in this case 6 seconds)

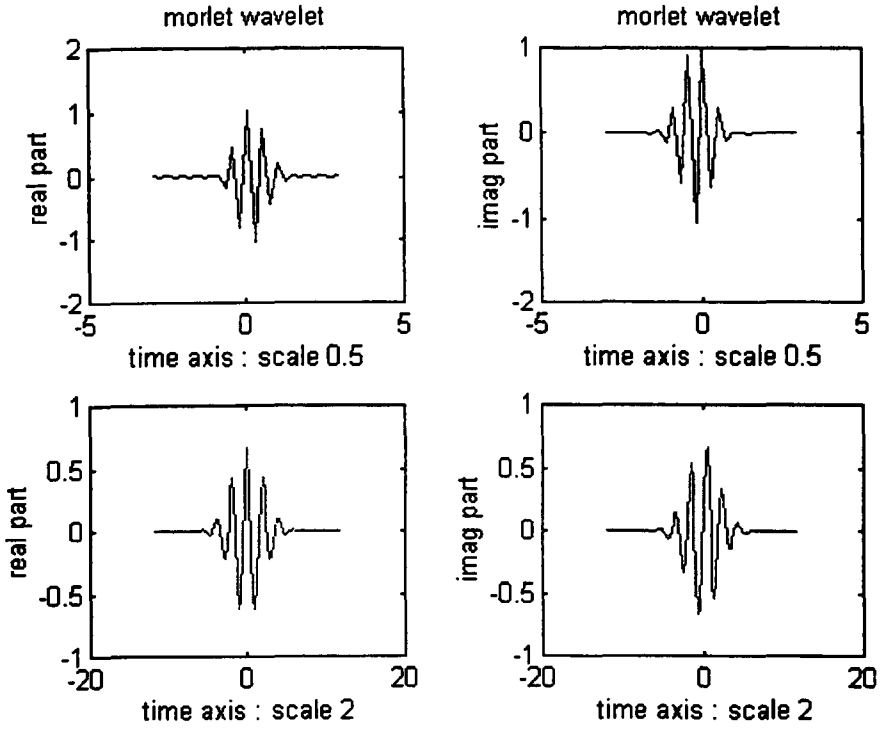


Fig.5.7 Real and imaginary part of Morlet wavelet at scales 0.5 and 2.

In a few examples we consider first a square wave, sampled at a frequency of 4 Hz (t_s=.25 sec) and with 1024 samples. We will use the dilation of the Morlet wavelet to detect the increasing accuracy (read : resolution) of the position of transients. As a second function we consider a chirped signal (frequency modulated signal) also sampled at 4 Hz with a total length of 256 seconds and a frequency changing between 0 and 2 Hz. In this case, the instantaneous frequency will be detected. Fig. 5.8.

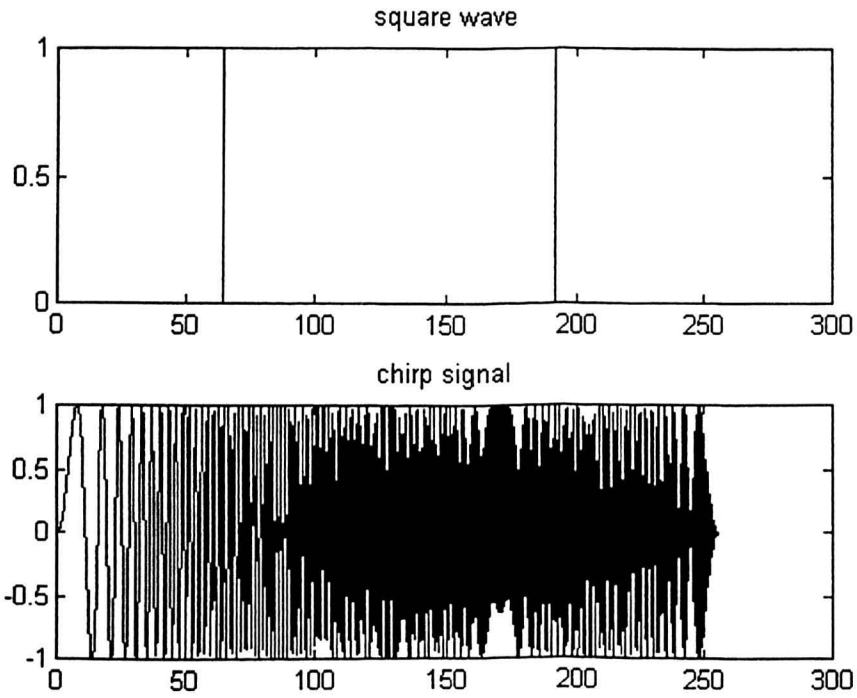


Fig.5.8. Square wave and chirped signal as test signals for Morlet transform.

The amplitude of the Morlet transformed square wave results in a 3D construction which is in a decreasing scale (narrower wavelet) with an increasing accuracy pointing to the transient points on the time axis. The square wave starts at $t_1 = 64$ seconds and stops at $t_2 = 192$ seconds. At scale .5 the wavelet is only 6 seconds wide and the result of the inner product operations shows clearly some peaking at t_1 and t_2. (Fig. 5.9) An even better impression is acquired with a 2D plot where the amplitude is presented by a grey scale. (Fig. 5.10) The amplitude of the Morlet transformed chirped signal doesn't, at first glance, produce a clear picture. The 2D plot shows how the frequency changes with time. Remark the detection of the 1Hz frequency at scale 1 in the middle of the time scale. 2 Hz is detected at the end of the signal at scale 0.5.

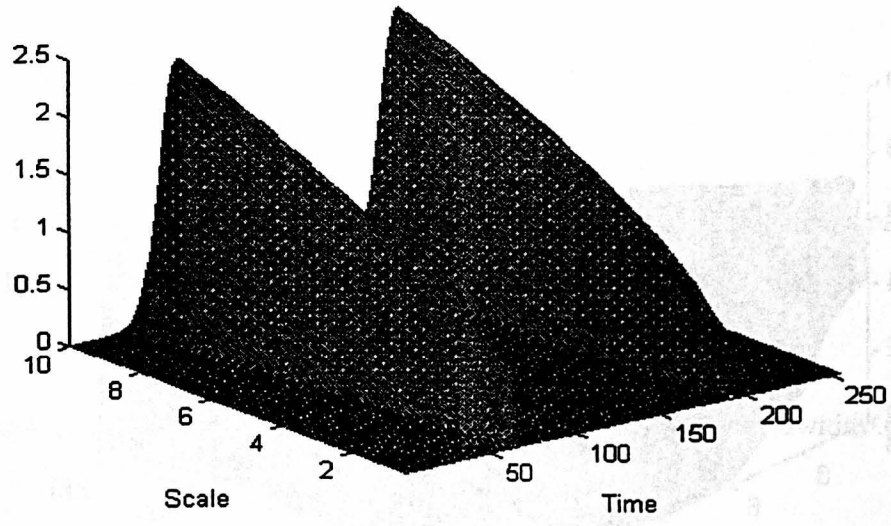


Fig. 5.9. 3D amplitude representation of square wave scalogram. High scale (=low frequencies) contains large but very inaccurate information about time events. Only at low scales (=high frequencies) the accurate time information when transients take place are revealed.

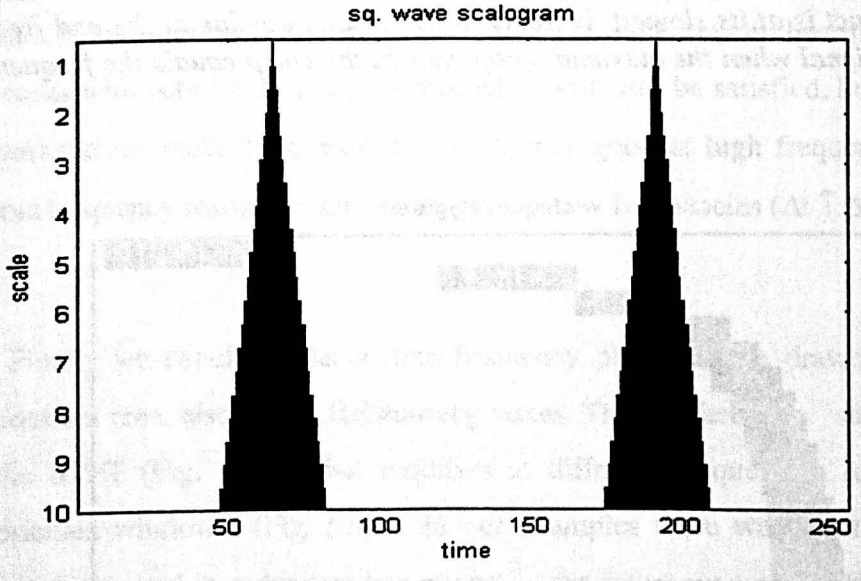


Fig. 5.10. 2D amplitude representation of square wave scalogram. Remark how well the transients events are detected at low scales (=high frequencies).

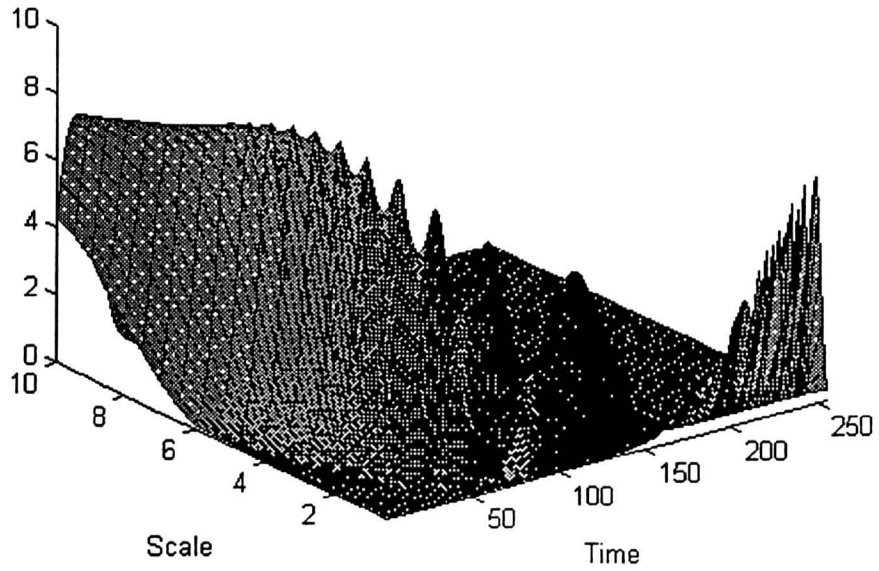


Fig.5.11. 3D amplitude representation of chirped signal. At high scales the wavelet detects only a few frequency. At low scales the wavelet is short and the inner product quickly increases, reaches a maximum and decreases as the frequency of the chirp signal changes. At scale 1 the frequency accuracy is at its lowest so the change of the inner product is at its slowest. It reaches its maximum value at the end frequency of the chirp signal when the maximum frequency in the chirp equals the frequency in the wavelet.

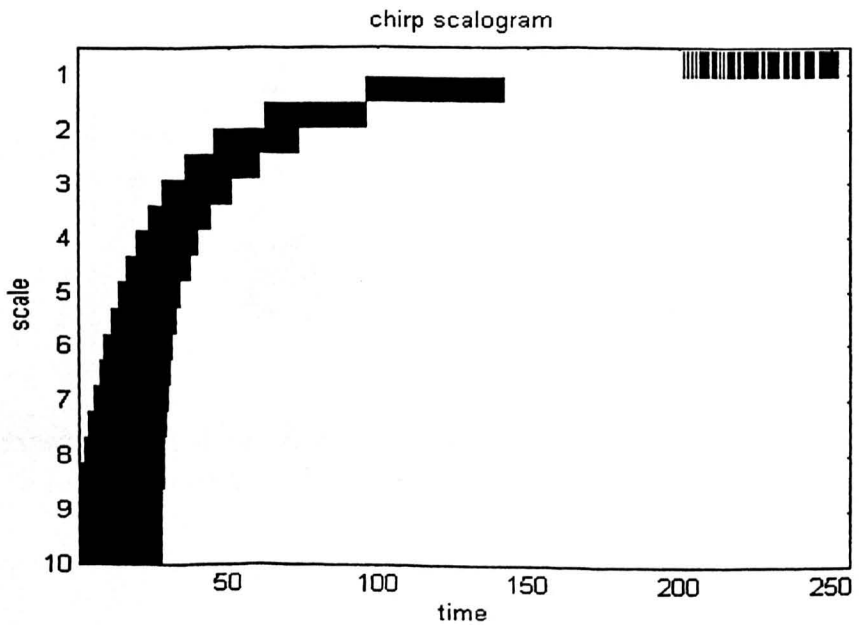


Fig.5.12. 2D scalogram of a chirp signal. The signal analysis starts in the bottom left and ends in the upper corner at the right.

5.4.3 Some Conclusions

- time resolution increases while incrementing the centre frequency of the analysis filters.

- $\Delta f/f$ is a constant, so these filters are off constant relative bandwidth .(Fig. 5.6)

- The main difference is that the STFT is linearly spaced in time and frequency domain and keeps the same window shape for all analysis. The Morlet wavelet , however changes the window shape at different scales, it still is linearly spaced in time and frequency (see Fig.5. 11,12) but will in the future be modified to a logarithmically scaled frequency axis . This gives great advantages for fast analysis and comes very close to natural phenomena like the human hearing system which works with octaves. Heisenberg inequality will still be satisfied, but now it will be possible to make time resolution arbitrarily good at high frequencies ($\Delta t \downarrow \Delta f \uparrow$) and frequency resolution arbitrarily good at low frequencies ($\Delta t \uparrow \Delta f \downarrow$).

- Finally we conclude that a time frequency plane can be drawn with boxes of constant area, also called **Heisenberg boxes**. The surface area stays the same for the STFT (Fig. 5.4.a) but modifies at different frequencies for the wavelet oriented window . (Fig. 5.13.). In our examples there was a lot of overlapping which resulted in redundant operations. In the future we will try to reduce or even eliminate that redundancy.

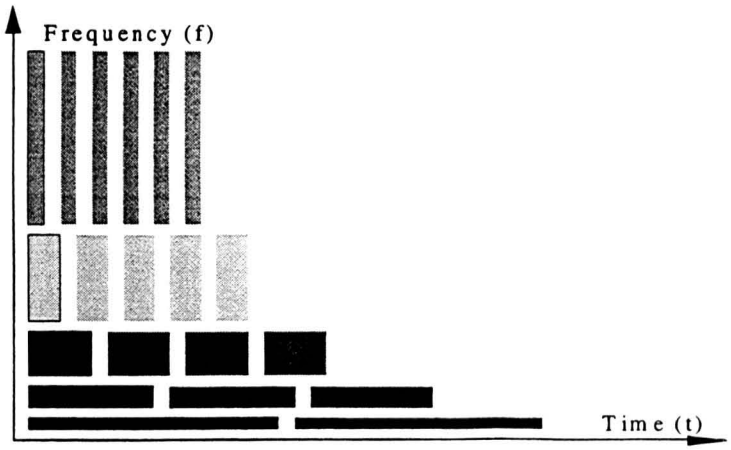


Fig. 5.13. Multiresolutional representation of Heisenberg boxes. With the orthogonality requirements fulfilled the boxes would stick together and realise a perfect overlapping of the time-frequency plane.

5.4.3 Wavelets

Generalising the concept of changing resolution at different frequencies leads us to the **continuous wavelet transform (CWT)** where all the impulse responses for the filter banks are defined as scaled versions of the same prototype $\psi(t)$, called basic wavelet.

$$\psi_a = \frac{\psi(t/a)}{\sqrt{|a|}} \tag{5.16}$$

$a \in \Re$ is the **scale factor** and acts in a similar way as the scale factor on geographical maps. With $a > 1$ the function is expanded, with $0<a< 1$ the function is contracted. Translated into map terms : the larger the scale factor the more global the view, the smaller the scale factor the more the detail in a small area is disclosed . The constant $1/\sqrt{|a|}$ stands for energy normalisation. The scale factor is the inverse of frequency. The Eq. (5.16) would indeed look very much similar to expression (5.15) if we would have put 1/a in stead off f in it. This modification to the Gabor transform lead us to the **Morlet wavelet** used in last paragraph. For wavelets, in general, the impulse response is not restricted to a modulated Gaussian window therefore the use of a general

expression $\psi(t)$. Considering all this we can come to a definition of the continuous wavelet transform (CWT) :

$$CWT_x(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t)\psi(\frac{t-b}{a})dt \qquad (5.17)$$

Note that the window sliding parameter τ has changed into b.(a and b $\in \Re$.)

Up till now, we proclaimed the engineer's viewpoint. It allowed us to intuitively understand how filter banks could be described and used for, not an ideal but an 'as lossless as possible' analysis.

To create wavelets for a specific task like orthogonal decomposition, severe constraints will have to be put on $\psi_{a,b}(t)$ and a more accurate mathematical approach will be necessary.

First some definitions : Eq. (5.17) can be seen as an inner product which measures the 'similarity' between the signal x(t) and the wavelet function $\psi(t)$.

$$CWT_x(a,b) = \langle x, \psi_{a,b} \rangle \qquad (5.18)$$

Similar as in the STFT where the continuing basis functions ($e^{j2\pi ft}$) from the Fourier transform are replaced by local oscillatory basis functions shifted in time and frequency with the parameters τ and ϕ the CWT will use b and a as parameters for the respectively time shift and scaling (= the inverse of frequency) . **The analysis results in a set of coefficients which indicate how close parts of the signal are to the dilated (with parameter a) and translated (with parameter b) version of the wavelet basis function.**

It would be very convenient if these dilated and translated versions of the wavelet form an orthogonal basis so that the wavelet transform, as the classical Fourier transform, results in an orthogonal decomposition.

Very interesting is the investigation of what is going to happen with the resolution boxes now that we are going to work with a multiresolutional analysis. Considering the possibility that $\psi_{a,b}$ could have an asymmetric shape, the expression (3) and (4) should be generalised and next to the resolution (Δt) the centre (t_0) of the wavelet should be defined. ($t_0 = 0$ for symmetric basic functions)

$$t_0 = \frac{\int_{-\infty}^{\infty} t(|\psi(t)|)^2 dt}{\int_{-\infty}^{\infty} (|\psi(t)|)^2 dt} \tag{5.19}$$

$$\Delta t^2 = \frac{\int_{-\infty}^{\infty} (t - t_0)^2 (|\psi(t)|)^2 dt}{\int_{-\infty}^{\infty} (|\psi(t)|)^2 dt} \tag{5.20}$$

The same goes for the frequency domain where f_0 will stand for the centre of FT(ψ) and Δf stays of course the frequency resolution. They can be calculated with similar Eq. as (5.12) and (5.13). The resulting intervals where data is collected are :

- for the time domain : $[b + a.(t_0 - \Delta t) , b + a.(t_0 + \Delta t)]$
- for the frequency domain : $[(f_0 - \Delta f)/a , (f_0 + \Delta f)/a]$

Their product results, as for the STFT (already shown in Fig. 5.1), in resolution rectangles with constant area $= 4.\Delta t.\Delta f$. Their position, width and height are however determined by a and b which must take discrete values. A perfect matching of the Heisenberg boxes in the time- frequency plane is our goal. This will be realised with orthogonal wavelets.

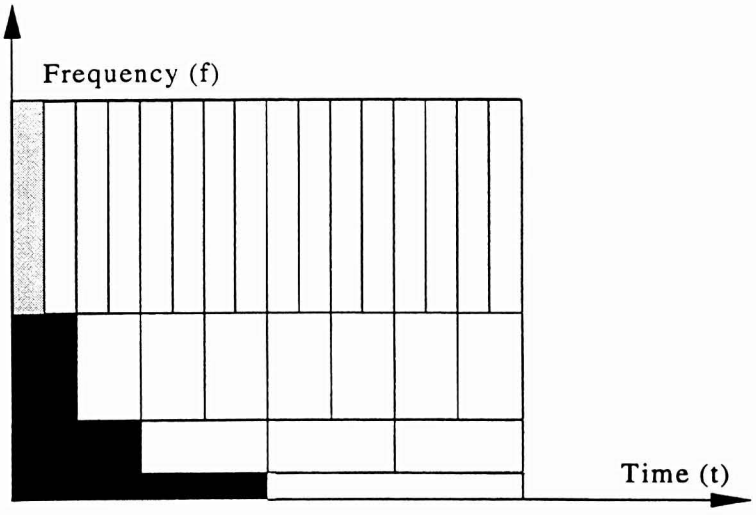


Fig. 5.14. The perfect match : multiresolutional analysis with orthogonal wavelets. In this case a and b are already discretized at powers of 2.With arbitrary values of a &b there will be redundancy.

It would be nice to have a reconstruction scheme complementary to the decomposition one. The condition for the existence of the inverse transform is :

$$C_\psi = \int_{-\infty}^{\infty} \frac{(|FT(\psi)|)^2}{f} df < \infty \qquad (5.21)$$

Remark that $C_\psi < \infty$ implies (FT(ψ) at f = 0) = 0 = $\int_{-\infty}^{\infty} \psi(t)d(t)$. Or, $\psi(t)$ has zero mean and it has to oscillate. This, together with the final existence property has given ψ the name of 'small wave ' (in French : **ondelette**) or in English : wavelet.

The inverse transform , taking into account that a > 0 , is given by

$$x(t) = \frac{1}{a^2 C_\psi} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} X(a,b)\psi_{a,b}(t)dbda \qquad (5.22)$$

Without special constraints, $\{\psi_{a,b}(t)\}$ are certainly not orthogonal since a and b are still considered as real values and the analysis is very much redundant. Frames and

B/A ratios for specific values of a,b are now to be considered to achieve information on good reproducibility. [57]

An at first glance contradictory comment on whether or not striving to orthogonality comes from J. Morlet : He states that it can be interesting to pursue redundancy ! It allows the wavelet coefficients to be stored at low-bit resolution (2 bits) while still being able to reconstruct the signal with comparatively much higher precision.

5.4.4 Multiresolutional Analysis

It seems that just observing and studying the resolutions in the time-frequency plane cannot guide us to a perfect lossless analysis. Maybe a more abstract study could lead us to a deeper understanding and a wider approach of the principle of mutiresolutional analysis. Instead of just trying to split up the content of signals into well defined frequency band components covering the whole signal why not examine signals in a hierarchical set of subspaces and then putting very specific constraints on them so they would become exclusive and consequently contain unique information about the signal. A compact way of describing multiresolution analysis is by defining a sequence of subspaces V_j of $L^2(\Re)$, $j \in \mathbf{Z}$, with the following properties :

1. $V_j \subset V_{j+1} \ldots \subset L^2(\Re)$
2. $x(t) \in V_j \Leftrightarrow x(2t) \in V_{j+1}$
3. $x(t) \in V_0 \Leftrightarrow [\{x(t-k) \mid k \in \mathbf{Z}\}] \in V_0$
4. $\cup V_j \mid \forall j = L^2(\Re)$ and $\cap V_j \mid \forall j = \{0\}$
5. There is a function $\varphi(t)$ with $\int_{-\infty}^{\infty} \varphi(t)\, d(t) \neq 0$ such that $\{\varphi(t-k) \mid k \in Z\}$ is a Rietz basis for V_0.

A **Rietz** basis means that for $x(t) \in V_0$ there exist a unique sequence $\{\alpha_k\} \in l^2(Z)$ such that $x(t) = \sum_{k \in Z} \alpha_k \varphi(t-k)$.

Since $\varphi \in V_0 \subset V_1$, there is a sequence $\{h_k\} \in l^2(\mathbf{Z})$ such that $\varphi(t)$, called the scaling function satisfies :

$$\varphi(t) = 2\sum_k h_k \varphi(2t-k) \qquad (5.23)$$

This equation also written as an inner product : $\varphi(t) = 2\langle h_k , \varphi(2t - k\rangle$

It is known as the **dilation equation for the scaling function**. Other names are the **refinement equation** or also the **two-scale relation (TSR).** Fig. 5.15. illustrates the scaling operation with the block function.

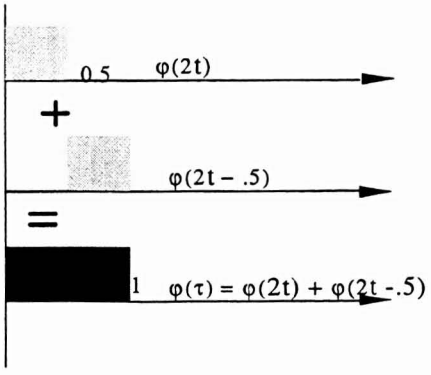


Fig. 5.15. Illustration of the dilation equation form.(16) for scaling function with the block function $\chi_{[0,\tau)}(t) = \{1 \mid 0 \le t < \tau, 0 \mid else\}$. $h_k = \{1/2, 1/2\}$

By normalising $\int_{-\infty}^{\infty}\varphi(t).dt = 1$ and integrating both sides of eq. (16) we obtain :

$$\sum_k h_k = 1 \qquad (5.24)$$

Mostly no explicit expression for φ is available. Solving the dilation equation for some choice of the coefficients h_k results in an expression where dilation and translation properties are incorporated :

$$\varphi_{j,k}(t) = 2^{j/2}\varphi(2^j t - k) \qquad (5.25)$$

If necessary there are fast algorithms to evaluate the scaling function φ at dyadic points $t=2^{j-1}.k$, $j,k \in \mathbf{Z}$. Most of the time however it is , only theoretically interesting know the scaling function. **h_k, however will frequently be used to go from one scale to another one.**

This seems very abstract and far away from the resolution defined up till now. With property 5 we stated that $\varphi(t)$ and its translates are a basis for V_0 . **We define now the reciprocal of the translation distance as the (time) resolution of the basis.** The resolution must now be understood as the number of basis functions per unit length. Normalising the resolution to be 1 in V_0 , it will accordingly be 2^j in V_j . Let us consider as an example the set of subspaces described as the 'sample and hold functions' in $L^2(\Re)$. For a specific function $x(t) \in L^2(\Re)$ (Fig. 5.8.) ,we can imagine that the piece wise linear function $x_j(t)$ in fig.b is $\in V_j$, in fact it is a projection of x(t) on φ_j . The other functions in fig.c and fig.d are also projection but in higher subspaces. The basis function for all these projections is the block function and its dilated versions : $\varphi(t) = \chi_{[0,\tau)}(t) = \{1 \mid 0 \le t < \tau\ ,\ 0 \mid \text{else}\}$

- with $\tau = 1$ in fig.b : resolution 1
- with $\tau = .5$ in fig.c : resolution 2
- with $\tau = .25$ in fig.d. : resolution 4

Remark the relation between the sample and hold time τ, and the resolution : The more samples taken (τ small) the more accurate the information about x(t) the larger (= better) the resolution .

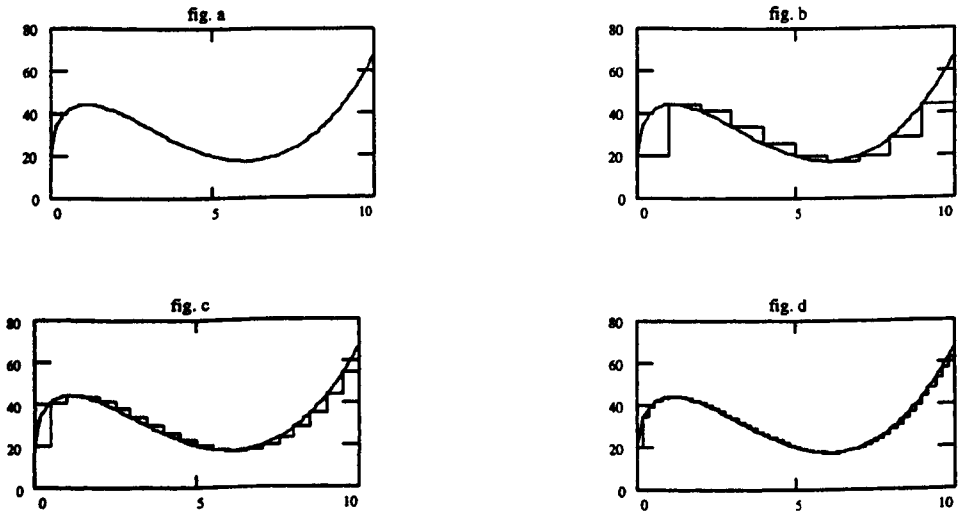


Fig. 5.16. The function in fig.a could stand for an arbitrary function $x(t) \in L^2(\Re)$. The others in fig .b,c,d are members of subspaces of $L^2(\Re)$. In fact they are projections of x(t) on the bases of the subspaces. The one in fig.b is from a 'lower' subspace of $L^2(\Re)$, the one in fig.d comes from a 'higher' subspace.

.

As $\varphi(t)$ is a basis and all its dilated versions are also bases, the dilation equation can not only be used to construct the scaling function in lower subspaces it is also applicable to construct arbitrary functions $x_{j-1}(t) \in V_{j-1} \subset V_j \subset L^2(\Re)$ as a linear construction with a basis in a higher subspace . The function x_{j-1} contains only half the number of coefficients per unit length as x_j (it is a blurred version)

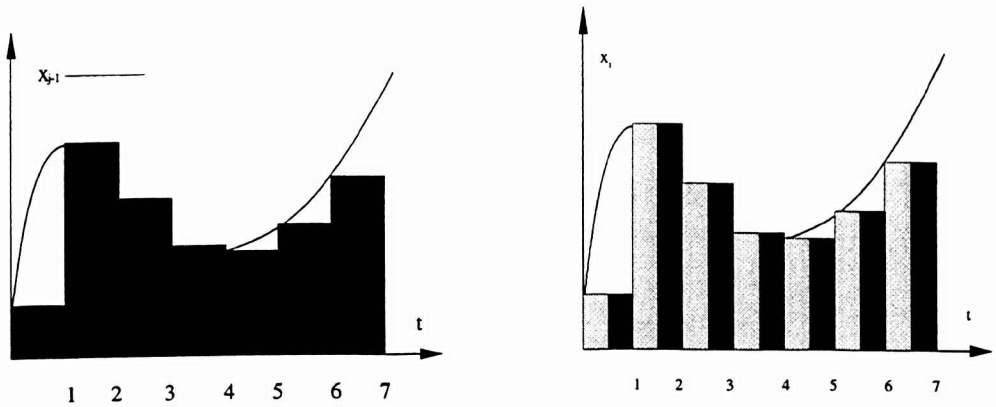


Fig. 5.17 . $x_{j-1}(t)$ at the left is constructed with the block function $\chi_{[0,1]}(t)$ as basis and at the right with a combination of $\chi_{[0,.5]}(t)$ and a delayed version of itself. It visualises how bases from higher subspaces can de used in a linear combination to represent functions out off lower subspaces Remark that this operation is possible at all subspace levels.

5.4.5 Orthogonal Wavelets

Let us now call W_j the complementary component of V_j in V_{j+1} such that :

$$V_{j+1} = V_j \oplus W_j \tag{5.26}$$

where the symbol $\oplus$ stands for direct sum . W_j is in fact the 'detail' to be added to V_j to obtain V_{j+1} . Remark that W_j is not unique and that there may be different ways to go from V_j to V_{j+1}. Iterating Eq .(5.26) results in :

$$V_{j+1} = W_j \oplus W_{j-1} \oplus W_{j-2} \oplus W_{j-3} \ \ldots. \tag{5.27}$$

Finally we must come to :

$$L^2(\Re) = \oplus W_j \,|\forall j \in Z \tag{5.28}$$

A function ψ **is a wavelet if** $\{ \psi(t-k) \mid k \in \mathbf{Z} \}$ **is a Rietz basis of** $\mathbf{W_0}$. As on page 125 for the scaling function we rewrite the definition of the Rietz basis now for the wavelet as : for $x(t) \in V_0$ there exist a unique sequence $\{\alpha_k\} \in l^2(Z)$ such that $x(t) = \sum_{k \in Z} \alpha_k \psi(t-k)$.

If so then consequently $\{\psi_{j,k} \mid j,k \in \mathbf{Z} \}$ is also a Rietz basis of $L^2(\Re)$. Note that there is much similarity in the definitions of φ and ψ. Without proofs we state by use of an inner product that

$$\psi(t) = 2 < g_k, \varphi(2t-k) > \qquad \text{or also}$$

$$\psi(t) = 2 \sum_k g_k \varphi(2t-k) \tag{5.29}$$

with similar operations as in Eq. (5.18) :

$$\sum_k g_k = 0 \tag{5.30}$$

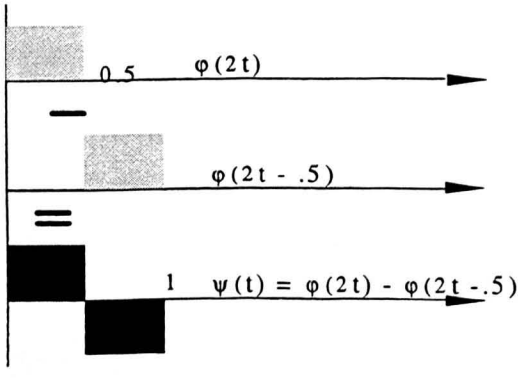


Fig. 5.18. Illustration of the dilation equation with the block function for wavelets. $g_k = \{1/2, -1/2\}$

As long as there is no orthogonality, **frames** are to be considered with wavelets as it was the case with the STFT. It could be that wavelet frames are just intermediate states in the research to orthogonality. However, as already said, redundancy can have his merits and frame investigation (B/A) based on specific choices of a,b and $\psi_{a,b}(t)$ can tell us about the redundancy in the analysis and the quality of reproduction. For

our applications they are only slightly more then of theoretical interest and we further restrict ourselves to referring to chapter 3 in a standard work on this field : "Ten Lectures on Wavelets " by Ingrid Daubechies. [57] .

An orthogonal multiresolution is defined when W_j is the orthogonal component of V_j in V_{j+1}. So, all spaces W_j are mutually orthogonal. This leads to some very interesting properties for scaling function, wavelet and filter coefficients h_k and g_k .

Equivalent to noting that $V_j \perp V_{j-m}$ (m=k-k') one can compactly write :

$$\langle \varphi_{j,k} , \varphi_{j,k'} \rangle = \delta_{k,k'} \quad k, k', j \in \mathbf{Z} \quad \delta_{k,k'} = 1 \text{ if } k=k', \; 0 \text{ else} \qquad (5.31)$$

Similar to the former expression one can formulate for $V_j \perp W_j$ and $W_j \perp W_{j'}$ with $j \neq j'$ that :

$$\langle \psi_{j,k} , \psi_{j',k'} \rangle = \delta_{j,j'}.\delta_{k,k'} \qquad k, k', j, j' \in \mathbf{Z} \qquad (5.32)$$

(5.31) and (5.32) together with the dilation equations (5.23) and (5.29) leads us to solutions for the filter coefficients h_k and g_k :

$$h_{k-2m} = \langle \varphi(t-m) , \varphi(2t-k) \rangle \quad \text{and} \quad g_{k-2m} = \langle \psi(t-m) , \varphi(2t-k) \rangle \qquad (5.33)$$

These are very general solutions for the shifted scaling function $\varphi(t-m)$. To calculate h_k and g_k just make m = 0 in eq. (5.33) . One can also find that deriving an orthogonal wavelet from an orthogonal scaling function results in the following relation between h_k and g_k :

$$g_k = (-1)^k . h_{1-k} \qquad (5.34)$$

For the block function this results in :

- for h_k : {1/2, 1/2}
- for g_k : {1/2,-1/2}

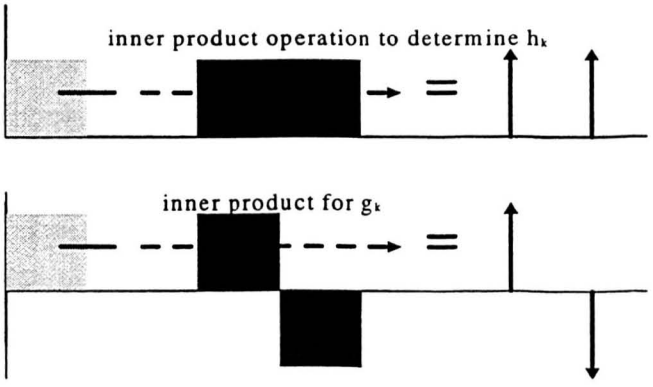


Fig. 5.19. Illustration of the determination of the coefficients h_k and g_k.

To complete the operation one needs a reconstruction formula as well. In particular for $\varphi(2x - k) \in V_1$ the reconstruction is produced with $\varphi(x) \in V_0$ and $\psi(x) \in W_0$:

$$\varphi(2t-k) = \sum_m h_{k-2m}\varphi(t-m) + \sum_m g_{k-2m}\psi(t-m) \qquad (5.35)$$

Sweldens [11] goes very much into detail on the mathematical background of all these expressions. It is advisable as a basis article and a future vision on wavelets and multiresolutional analysis.

The block function presented up till now as the scaling function was transformed into a wavelet with Eq. (5.29). One can find out now that this function, called the **Haar wavelet** has a lot of fascinating properties extensively described in literature. [66] We will use it in the future as our most primitive basis for analysis. Of course, $L^2(\Re)$ can be subdivided in different subspaces V_n with their details W_n . The work of **Daubechies** , **Coifman** and so many others [65] has lead us to a collection of wavelets that were investigated on their usability in digital signal processing.

5.4.6 Dyadic Sampling

As we are interested in a numerical analysis, let us find out, about the possible relation between the discretization of the time-scale parameters b,a and true orthogonal bases. A natural way to do so is to choose $a = a_0^j$ and $b = k.a_0^j.t_s$ with $j,k \in \mathbf{Z}$ and t_s = sampling time. This results in a **dyadic** sampling grid in the time scale plane . (Fig. 5.10). With a_0 close to 1 and t_s sufficiently small the wavelet will be over complete and the reconstruction we still be very close to (15) **With the computation performed in octaves (a_0=2) a true orthogonal basis will show up for very special choices of $\psi(t)$.** The **Haar wavelet** is one of them.

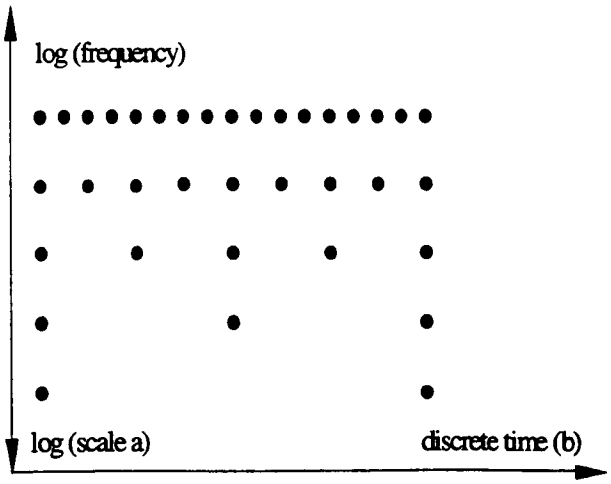


Fig. 5.20. Dyadic sampling grid in the time-scale plane. Each node represents a couple (k,j) . They are respectively used as integer multiples for the time shift a_0^j.ts and the exponent of the discrete scale a_0^j . For practical purpose we choose a_0 = 2, this leads to very simple decimation operations

5.4.7 Discretizing the Discrete Wavelet Transform

For a good choice of ψ we can now evolve from the CWT to the **Discrete Wavelet Transform (DCT)** with integer parameters j and k instead of the real values a and b :

with : $$\psi_{j,k}(t) = a_0^{-j/2} h(a_0^{-j} t - k t_s) \tag{5.36}$$

$$DWT_x(j,k) = \langle x(t) , \psi_{j,k}(t) \rangle \tag{5.37}$$

With a short hand notation one writes the wavelet coefficients of a function x(t) as $d_{j,k} = DWT_x(j,k)$. These are called the **detail** in the different spaces.

An arbitrary signal can now be represented as a weighted sum of the orthogonal basis functions :

$$x(t) = \sum_j \sum_k d_{j,k} \psi_{j,k} \tag{5.38}$$

This was a result never reached with STFT . Daubechies even proved that it **was impossible to have orthogonal bases** with well localised functions as the ones used in the STFT. [57] .

A next step in the discretization of the wavelet transform is the sampling of the analysing signal with sampling frequency F_s ($1/t_s$) . In fact, we want to come now to an algorithm for wavelet decomposition and reconstruction with discrete signals. Let $x_j \in V_j$. Then because of $Vj = V_{j-1} \oplus W_{j-1}$, x_j can be decomposed as :

$$x_j = x_{j-1} + y_{j-1} \quad \text{with} \quad x_{j-1} \in V_{j-1} \text{ and } y_{j-1} \in W_{j-1} \tag{5.39}$$

Iterating this process by use of form. (20) we obtain :

$$x_j = y_{j-1} + y_{j-2} + \ldots + y_{j-m} + x_{j-m} \quad \text{with} \quad m \in N \tag{5.40}$$

with : $$x_j(t) = \sum_k \lambda_{j,k} \varphi_{j,k}(t) \quad \text{and} \quad y_j(t) = \sum_k \gamma_{j,k} \psi_{j,k}(t) \tag{5.41}$$

The functions $x_j(t)$ and $y_j(t)$ are now respectively projected on $\varphi_{j,k}(t)$ and $\psi_{j,k}(t)$ and the projection coefficients $\lambda_{j,k}$ and $\gamma_{j,k}$ are kept for further analysis. To go from

one scale another one has to cede along Mallat's decomposition and reconstruction scheme as in Fig. 5.12 and apply the dilation formulas on $\lambda_{j,k}$ and $\gamma_{j,k}$.

For decomposition :

$$\lambda_{j,k} = \sqrt{2}\sum_m \lambda_{j+1,k} h_{2k-m} \quad \text{and} \quad \gamma_{j,k} = \sqrt{2}\sum_m \lambda_{j+1,k} g_{2k-m} \tag{5.42}$$

For reconstruction :

$$\lambda_{j+1,k} = \sqrt{2}\sum_m \lambda_{j,k} h_{k-2m} + \sqrt{2}\sum_m \gamma_{j,k} g_{k-2m} \tag{5.43}$$

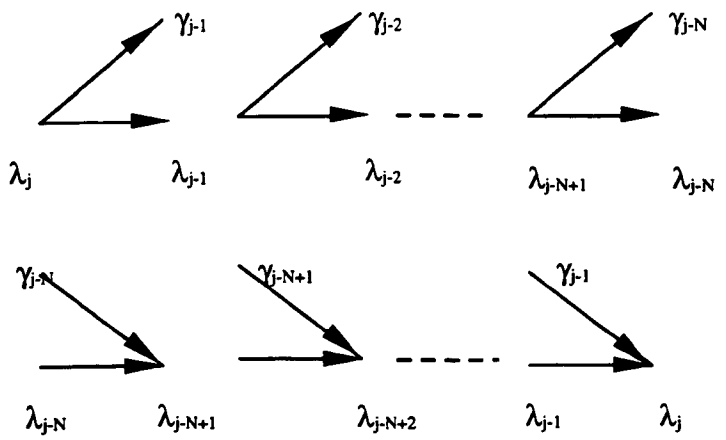


Fig. 5.21. Decomposition and reconstruction scheme for discrete values λ_n and γ_n. This is also called Mallat's Fast Wavelet algorithm.

Skipping the mathematical proofs [4] one can finally sample the continuous functions x(t) now becoming $x(n.t_s)$ or shortly written x(n) and apply the decomposition and reconstruction formulas on discrete data vectors. In fact we start with the values $c_{0,k}$ as the discrete values of a signal . If for instance $\varphi_0(t)$ was the sample and hold function one could easily understand that a continuous function $x(t) \in L^2(\Re)$ is scaled to a piece wise constant function $\in V_0$ by a projection on the basis (φ_0) of that space. We continue now our analysis with the discretized values = the sample values. Generalising this idea could comes from Eq. (5.35).

Substituting now $\lambda_{0,k}$ as the 0^{th} level of analysis by x_k ,Eq. (5.35) becomes

$$\lambda_{-1,k} = \sqrt{2}\sum_m x_k h_{2k-m} \quad \text{and} \quad \varphi_{-1,k} = \sqrt{2}\sum_m x_k g_{2k-m} \tag{5.44}$$

The final reconstruction is realised with

$$x_k = \sqrt{2}\sum_m \lambda_{-1,k} h_{k-2m} + \sqrt{2}\sum_m \gamma_{-1,k} g_{k-2m} \tag{5.45}$$

Further properties like the spectral behaviour of wavelets will be investigated later. We will apply now Eq. (5.37) and (5.38) in Mallat's scheme.

5.4.8 The Haar transform

We will investigate the Haar transform with the following set of functions :

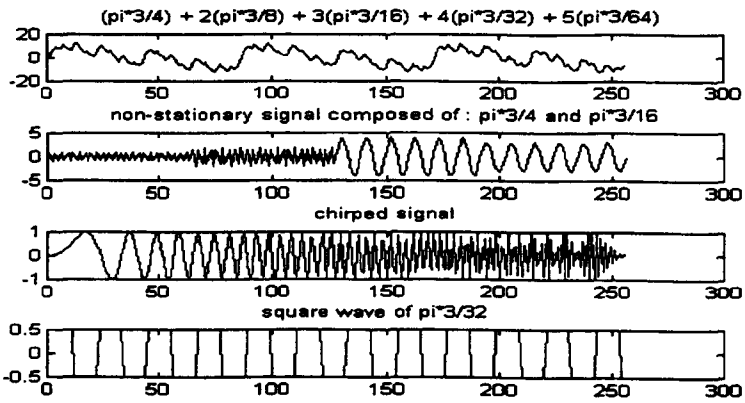


Fig.5.22. Four functions are considered to investigate the Haar wavelet : Signal 1 is a stationary signal composed of different sine waves proportionally decreasing in amplitude with increasing frequency and all well situated in the middle of the bandpass filters (3π/4 , 3π/8 , 3π/16 , 3π/32 , 3π/64) . Signal 2 is non-stationary and changes abruptly the amplitude from 1 to 2 for the frequency 3π/4 in the first part of the signal , in the second part of the signal the amplitude changes from 4 to 3 at a frequency of 3π/16. Signal 3 is a chirped signal, in fact a frequency modulated signal gradually changing between 0 and π. Signal 4 is a square wave at frequency π/16.

In the following analysis signal 1-4 will be wavelet transformed until level 3. This means implementing the operation explained in Fig. 5.21. At every level the detail (band pass filtering) and approximation (low pass filtering) coefficients are calculated

then decimation is performed. The next level starts with another splitting up of the former approximation coefficients in new details and approximation. The analysis goes on until level 3 where detail (picture 3) and approximation (picture 4) are finally kept. Remark that at the different levels a sample and hold action took place to artificially stabilise the results of the analysis at he same length and eliminate the decimation effect on the pictures (present the picture with its original frequencies). Although signal 1 was specially constructed to contain only 1 frequency / band pass , in fact per level (at $3\pi/4$, $3\pi/8$, $3\pi/16$, $3\pi/32$, $3\pi/64$) the analyses show combinations of different frequencies. This comes from the fact that , due to the very small amount of filter coefficients (for Haar : 2) , the roll off of the filter banks is very slow and frequencies from different bands, although attenuated, are also part of other bands. Remark the low frequency approximation in the last picture.

Fig.5.24 shows the absolute value of the FFT for the analyses shown in Fig. 5.23. Remark indeed the presence of different alias frequencies in the analysis. $3\pi/4$ is present at spectral line 96 in the first picture. The others are coming from frequencies out of other bands and of course distortion due to sample and hold operation disguises the analysis as well. In the second picture one can detect $3\pi/8$ at spectral line 48. An interesting experiment will be the use of higher order filter which will be the case when using Daubechies wavelets.(see 5.4.9)

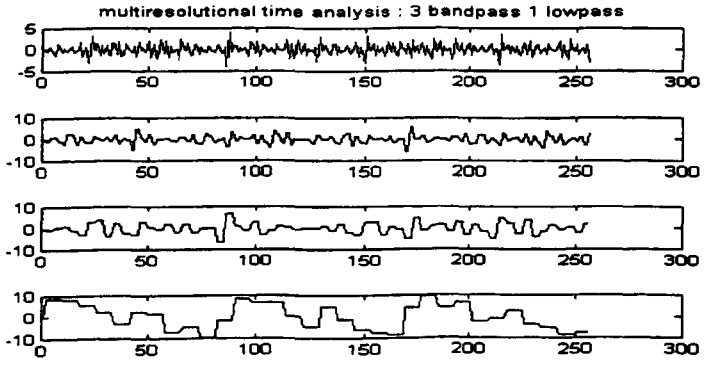


Fig.5.23. Signal 1 is respectively Haar band pass filtered between $\pi/4$ - $\pi/2$, $\pi/8$ - $\pi/4$ and $\pi/16$ - $\pi/8$. The results are called the detail coefficients at the levels 1, 2 and 3. Finally, the forth picture show the result of low pass filtering , they are called approximation coefficients.

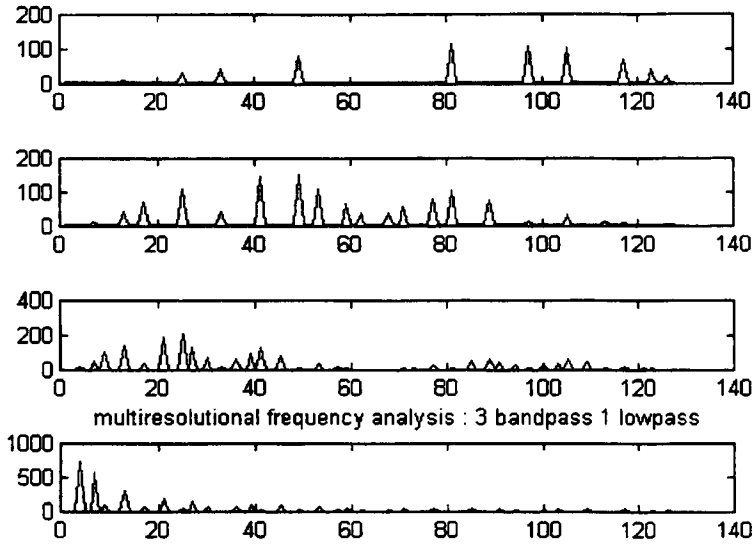


Fig.5.24. FFT of results in fig.5.23. The amplitude of the analyses are shown until π or spectral line 128. As the transition bands are very wide a lot of 'neighbouring frequencies' show up in the spectra of the different levels.

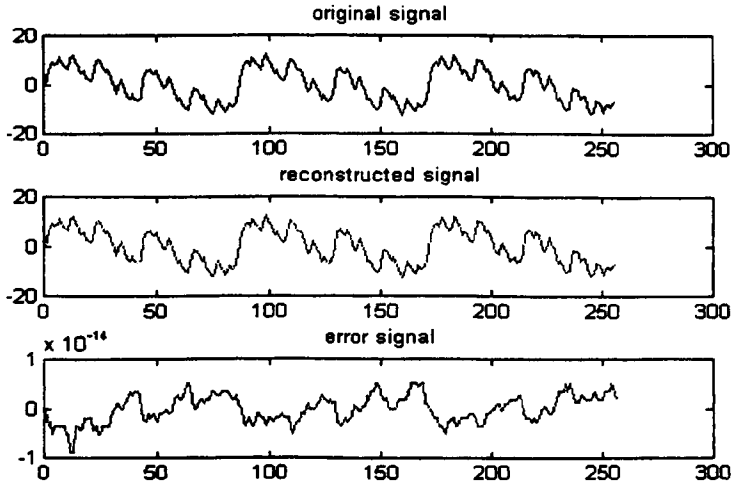


Fig.5.25. The original signal and the reconstructed one are compared and the error is calculated and shown.

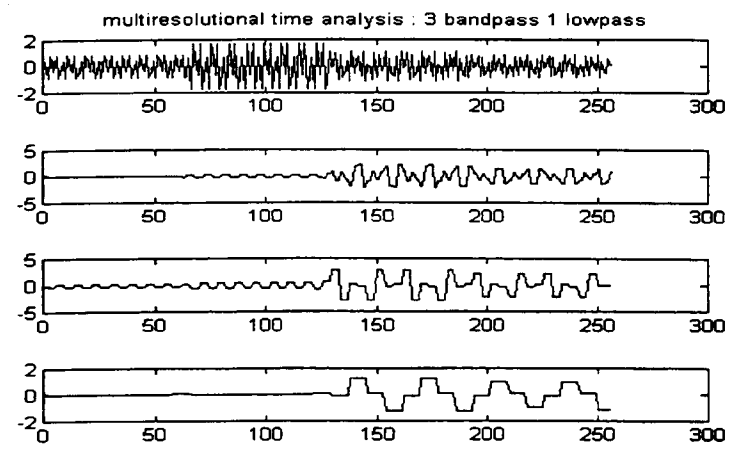


Fig.5.26. Two frequencies in different time slots are identifiable but not very well separated at the different levels.

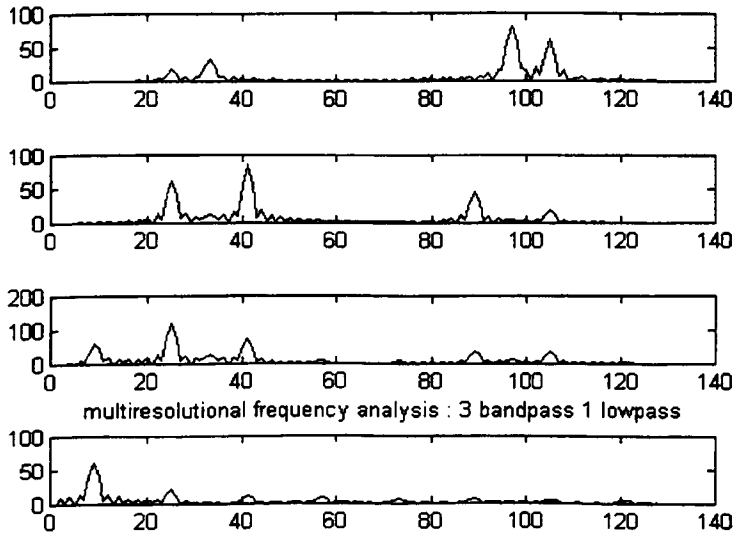


Fig.5.27. The frequency analysis shows that the 2 frequencies are not very separated. Wavelet analysis with more coefficients would result in a better separation of the signals at the different levels.

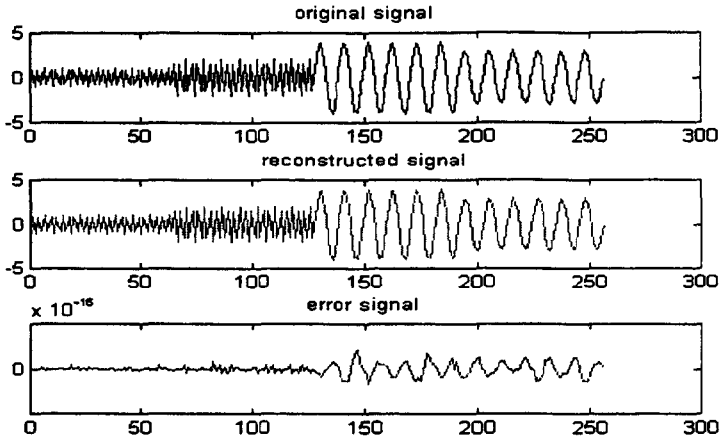


Fig.5.28. Error analysis of test function 2.

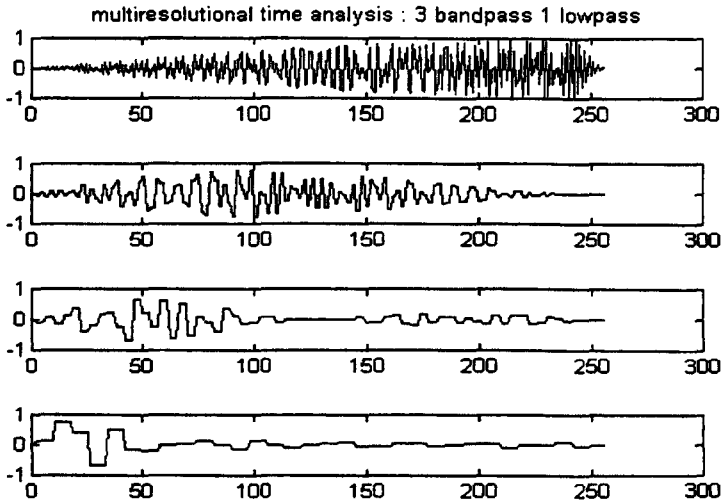


Fig.5.29. Haar wavelet analysis of chirp signal. As the frequency changes from the beginning until the end one can once again remark how big the transition regions in the filters really are.

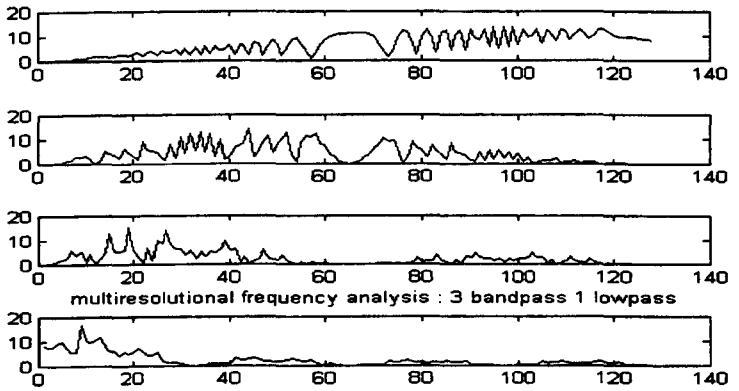


Fig.5.30. The spectral analysis extensively shows how slow the roll off of the Haar frequency acts.

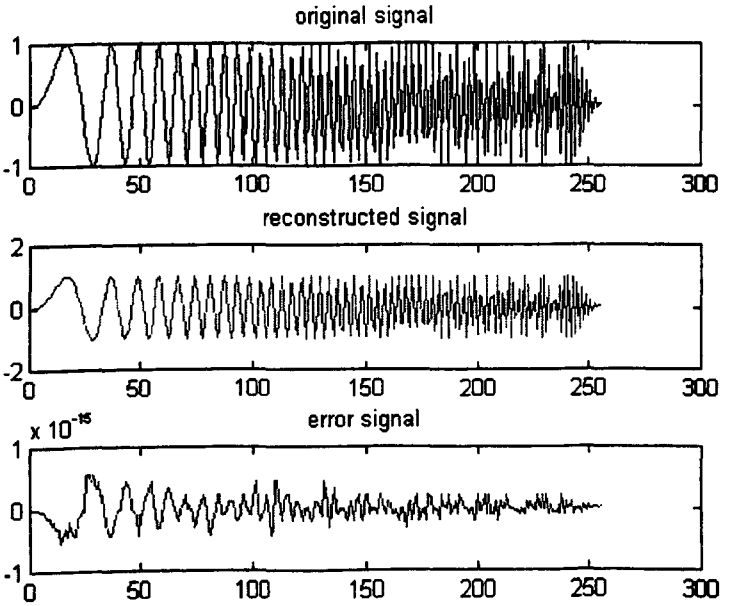


Fig.5.31. Error signal after Haar analysis and synthesis of chirp signal

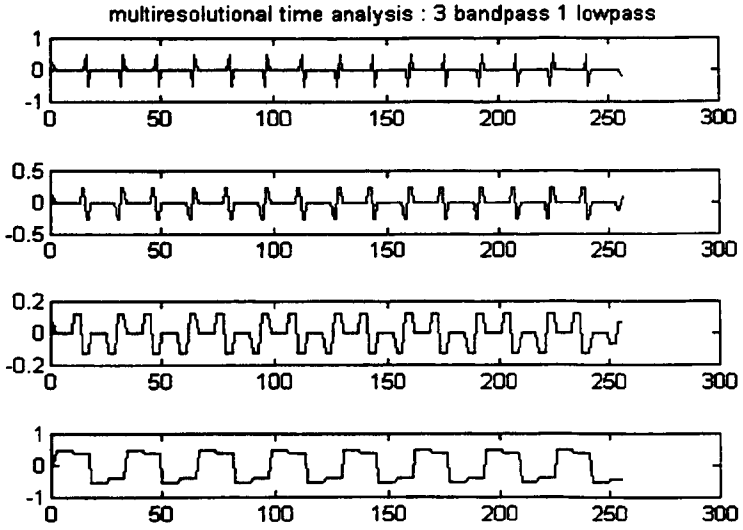


Fig.5.32. Haar wavelet analysis of square wave signal. Square waves are very similar to the Haar wavelet and thus can very efficiently be represented by it.

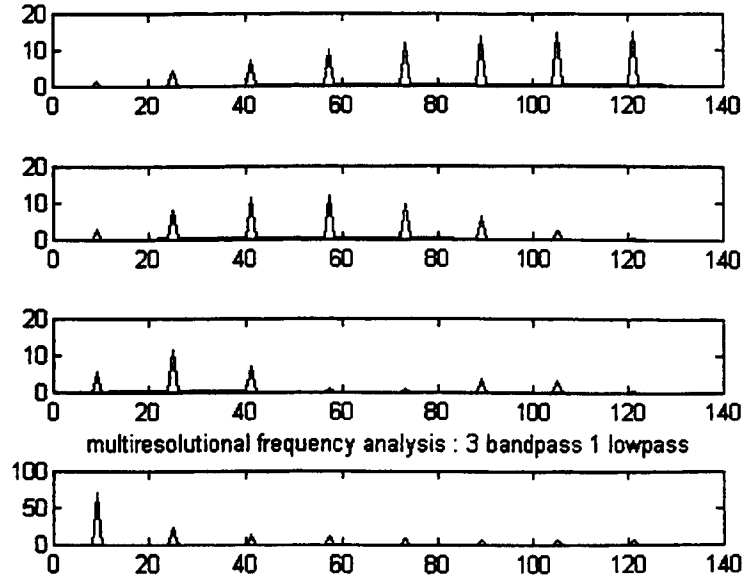


Fig.5.33. Spectral analysis of Haar transformed square wave. At index 8 the fundamental frequency is shown. The odd harmonics 3^{rd} 5^{th} 7^{th} etc. can be seen at indexes 24, 40, 48,...

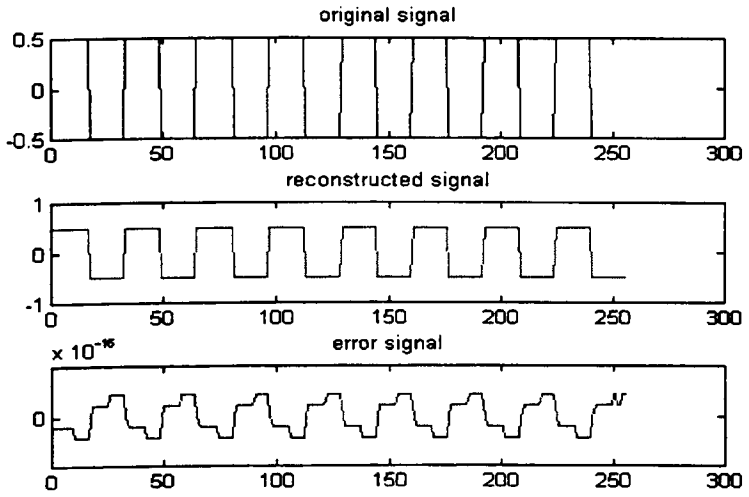


Fig.4.34. Error signal of Haar analysed and synthesised square wave.

5.4.9. Regularity

The Haar function, being a block function, shows some discontinuities which are mostly not present in the functions to be projected on the wavelet bases. This means that it is rather difficult to reconstruct the original signal with a limited number of Haar coefficients. Only 'edgy' functions or in fact edges in functions will be efficiently represented by Haar wavelets.

Intuitively one could feel that smoother wavelets would form 'better' bases to represent the majority of the functions. Better means : less coefficients to represent the function and smaller errors . We will look for more regularity (read differentiability) in the wavelet to more efficiently represent a function.

Regularity of a function can also be seen as a kind of smoothness. In fig.5.35 the daub4 and 16 scaling and wavelet functions iteration process is shown. Although they are both continuous, daub4 shows less regularity then the Daub16. Asking now for a smooth basis function , $\psi(t)$ will ensure us that no artificial discontinuities will appear in the transform coefficients or that every discontinuity in the transform coefficients is caused by the signal itself. Also, quantisation errors at a certain level in computing will after reconstruction, produce errors which are proportional to the basis functions

of that level. In image processing coding applications for instance, smooth error signals are less annoying to the eye than discontinuous ones.

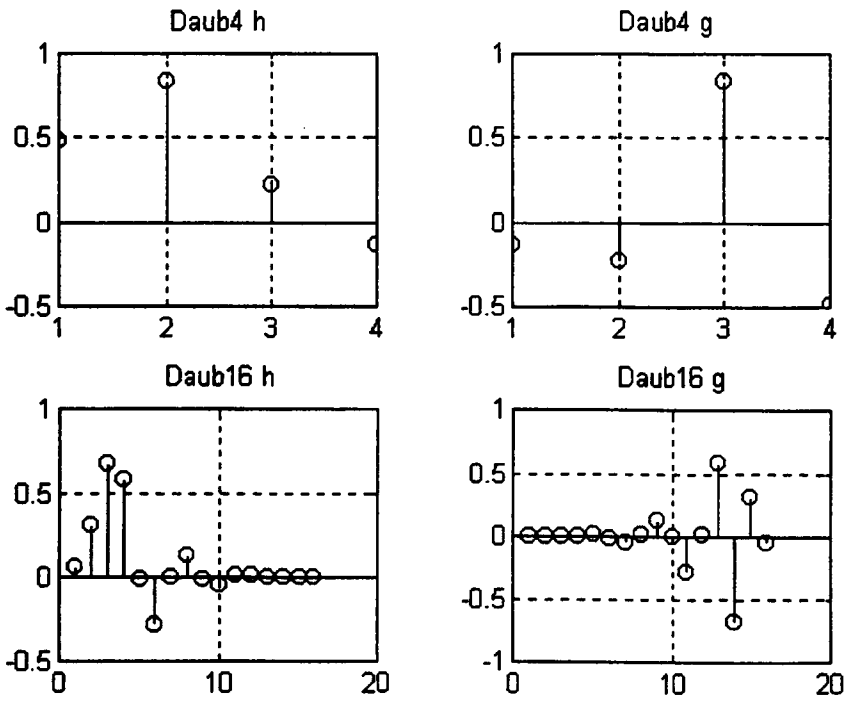


Figure 5.35. Daubechies scaling (h) and wavelet (g) filter coefficients.

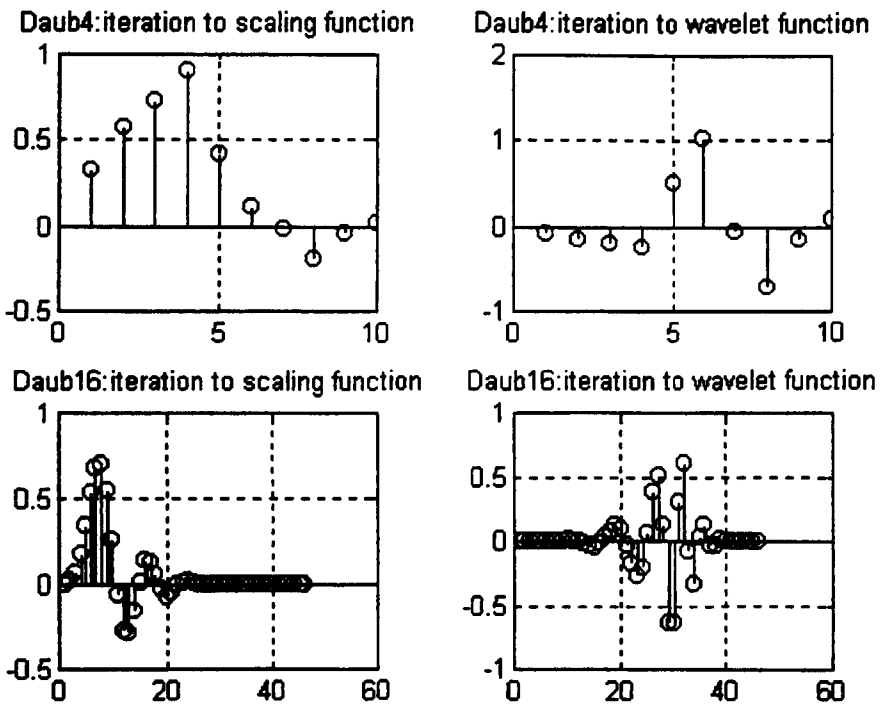


Figure 5.36. First iteration steps from filter coefficients to basis functions.

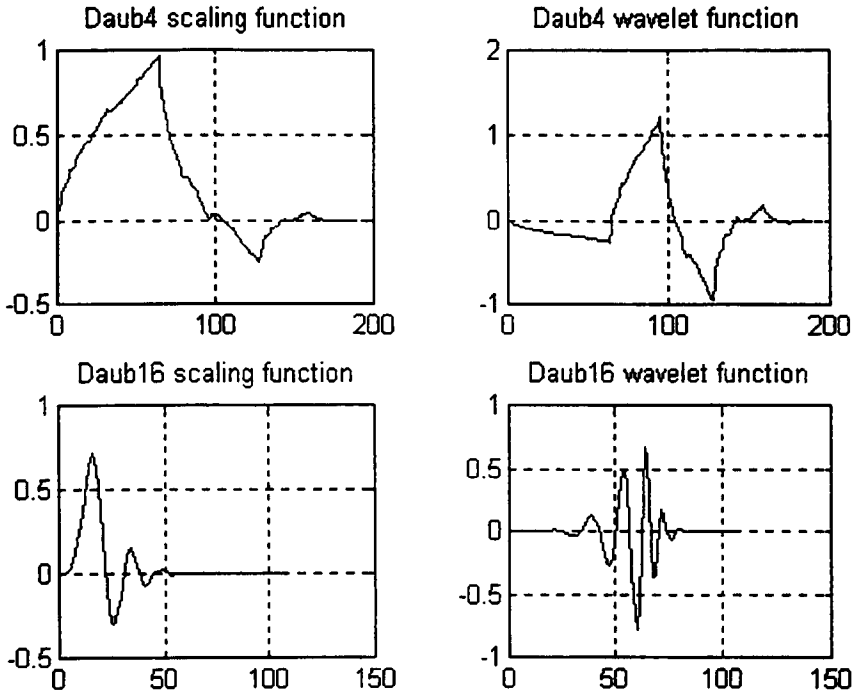


Figure 5.37. The iteration process on the filter coefficients must finally terminate in the scaling and wavelet functions.

Mathematically one can proof that there is a relation between :

$d^p\psi/dt^p$ = continuous function $\leftrightarrow$ $\Psi(\omega)$ has zeros at $\omega = 0$

or for the transfer function of the of the wavelet filter coefficients :

$G(e^{j\Omega})$ has poles at $\Omega=0$

One can even go a step further and state that the p^{th} derivatives of Ψ and G are zero at $\omega=0$ respectively $\Omega=0$. This results for Ψ in :

$$d^p\Psi / dt^p\big|_{\omega=0} = d^p / dt^p \int_{-\infty}^{\infty} \psi(t)e^{-j\omega t}\,dt\big|_{\omega=0} = (-j)^p \int_{-\infty}^{\infty} t^p\psi(t)dt = 0 \qquad (5.46)$$

This implies that the function $\psi(t)$ has P zero moments if $G(e^{j\Omega})$ has P zeros at $\Omega=0$. Analog to this deduction one comes to a condition for the wavelet filter coefficients:

$$\sum_{n=0}^{N-1} n^p g_n = 0 \quad \text{for p} = 0,1,\ldots. \text{P-1} \qquad (5.47)$$

This condition will be used in a condition set to generate Daubechies and other wavelets. (See appendix 5B)

5.4.10. Daubechies wavelets

The Haar wavelet is now seen as the first in a series of compactly supported wavelets designed by Ingrid Daubechies. The Haar wavelet is now also referenced as Daubechies2.

$n := 0..511$

$t6_{10} := 1 \quad t6_{511} := 0 \quad w_6 := \text{idwavelet}(t_6)$

$t5_{20} := 1 \quad t5_{511} := 0 \quad w_5 := \text{idwavelet}(t_5)$

$t4_{40} := 1 \quad t4_{511} := 0 \quad w_4 := \text{idwavelet}(t_4)$

$t3_{80} := 1 \quad t3_{511} := 0 \quad w_3 := \text{idwavelet}(t_3)$

$t2_{160} := 1 \quad t2_{511} := 0 \quad w_2 := \text{idwavelet}(t_2)$

$t1_{320} := 1 \quad t1_{511} := 0 \quad w_1 := \text{idwavelet}(t_1)$

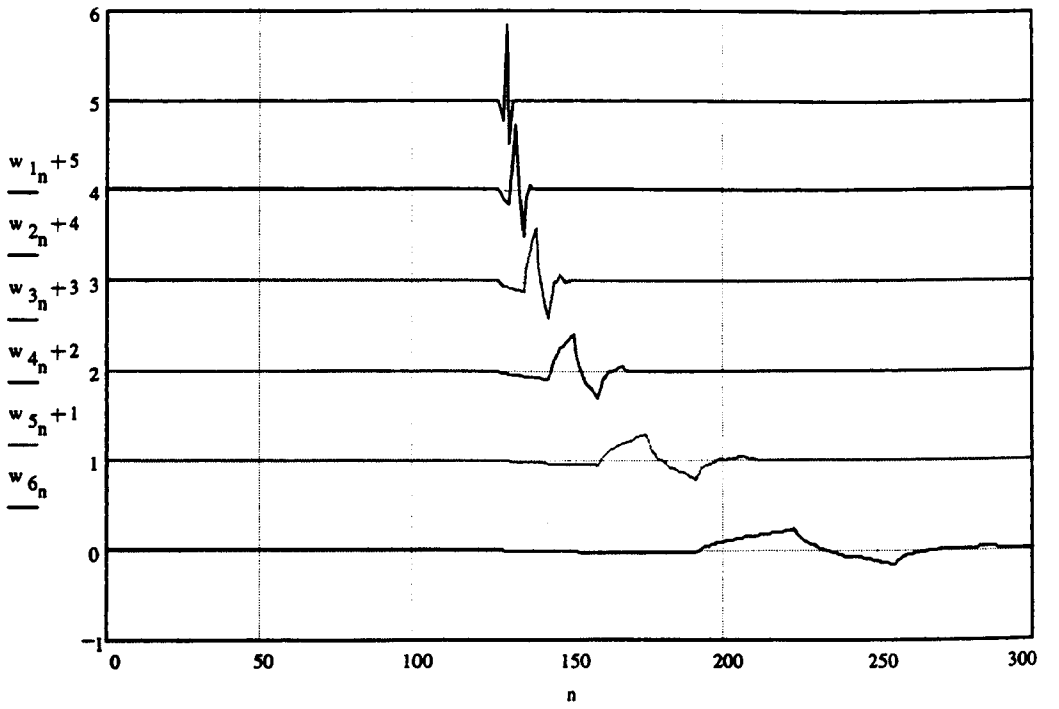


Fig.5.38. The Daub4 wavelet is reproduced from 6 different levels. They all start at index 128 but the higher the level the longer the wavelet: In a multiresolutional analysis one needs long wavelets for the low frequencies and short ones for the high frequencies.

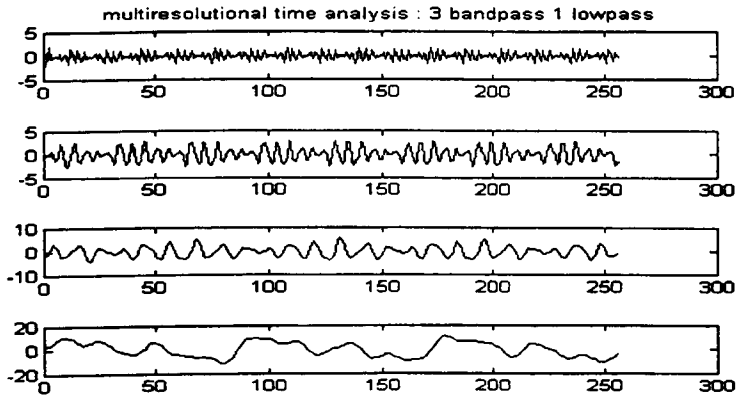


Fig.5.39. Daubechies8 analysis of 5 frequency signal. Remark how the different frequencies are better separated at the different levels

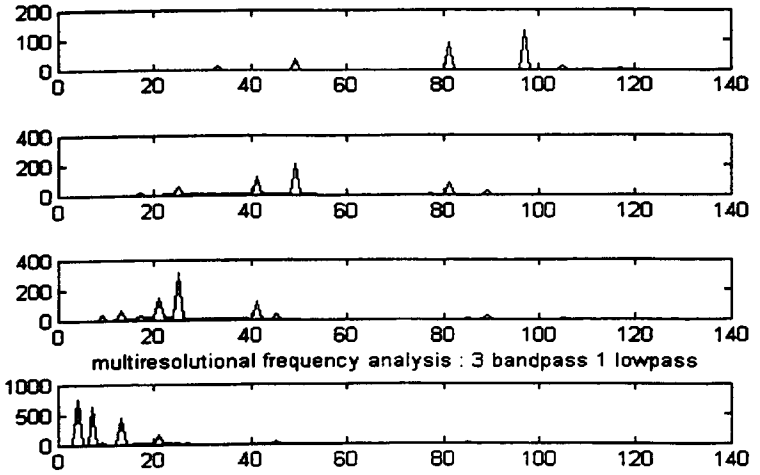


Fig.5.40. Spectral analysis of the Daubechies8 analysed test function 1. The spectral separation is much better than with the Haar transform.

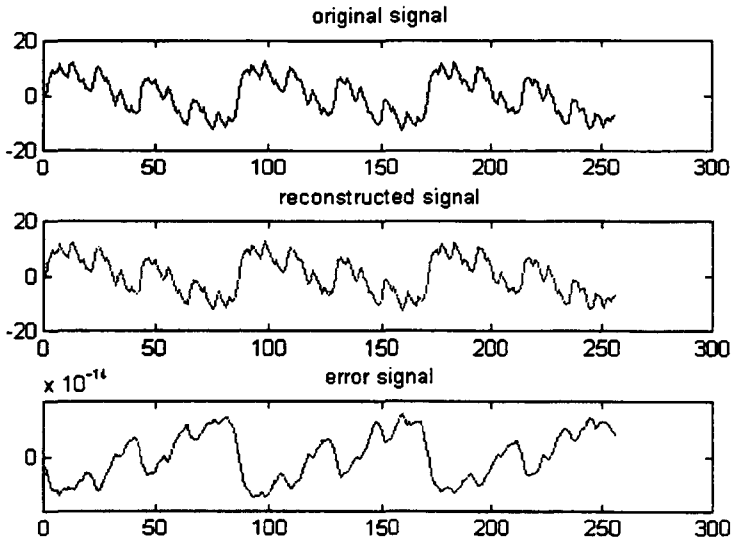


Fig.5.41. Error signal of Daubechies8 forwards and backwards transformed test function 1.

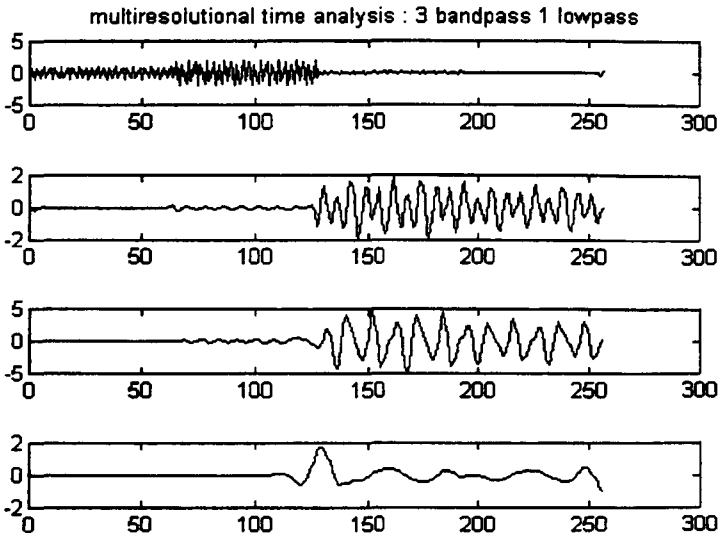


Fig.5.42. Daubechies8 analysis of 2 sine waves. As the signal is composed of the frequencies 3π/4 and 3π/16 Level 1 contains information about the former part of the signal at the frequency 3π/4. Level 3 contains information about the latter part of the signal at frequency 3π/16.

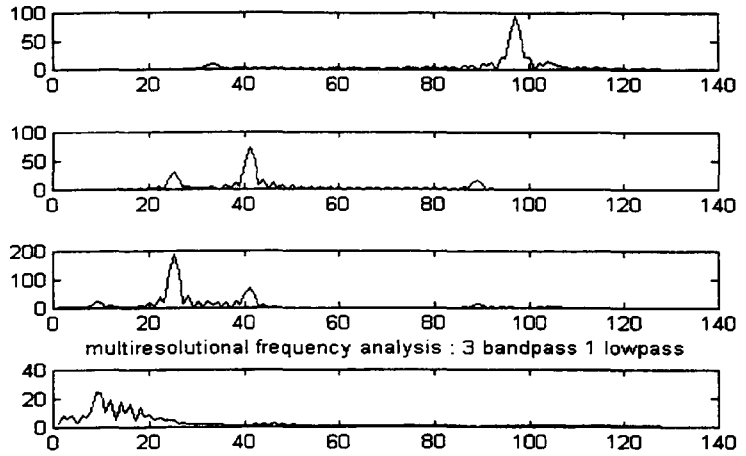


Fig.5.43. Spectral analysis of the 4 levels of a Baubechies8 analysed signal with 2 sine waves. (Test function 2).Remark the alias frequencies in the level 4.

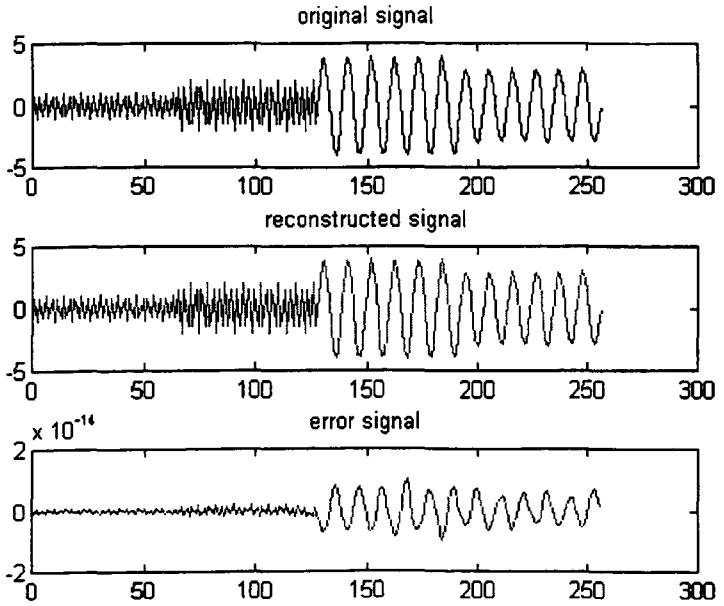


Fig.5.44. Error signal of Daubechies8 forwards and backwards transformed test function 2.

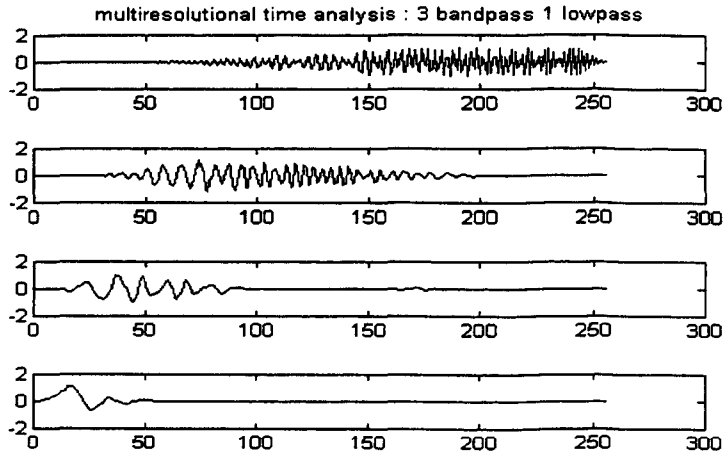


Fig.5.45. Daubechies8 analysis at 4 levels of test function 3. Remark the temporary character of the sine wave in the chirp signal : Level 1 shows the high frequencies at the end of the signal. Level 4 shows the low frequencies at the beginning of the signal

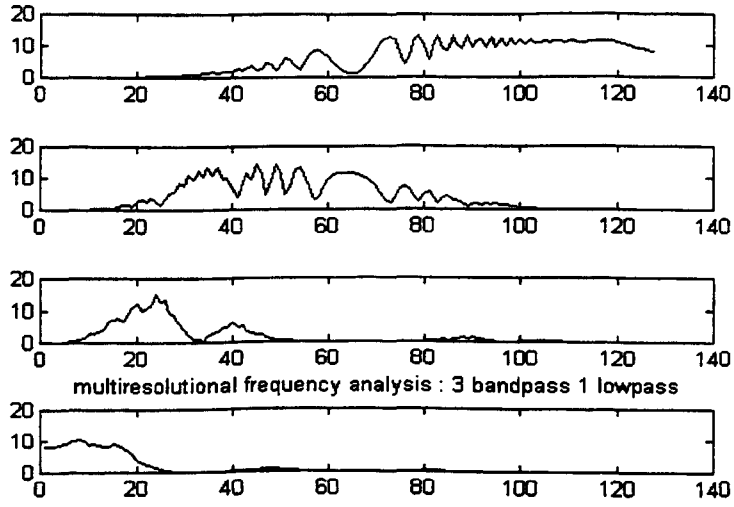


Fig.5.46. Spectral analysis of Daubechies8 analysed test function 3. 3 band pass levels and 1 low pass. As the chirp signal contains all frequencies, the low pass and the band pass filtering at different levels can very clearly be identified.

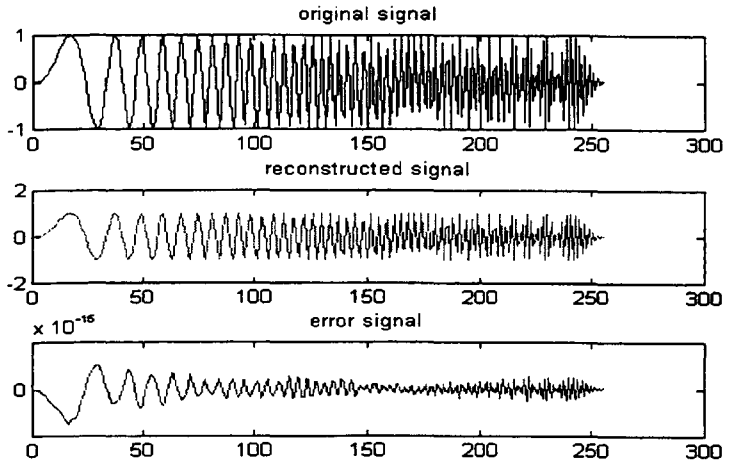


Fig.5.47. Error signal of Daubechies8 forwards and backwards transformed test function 3.

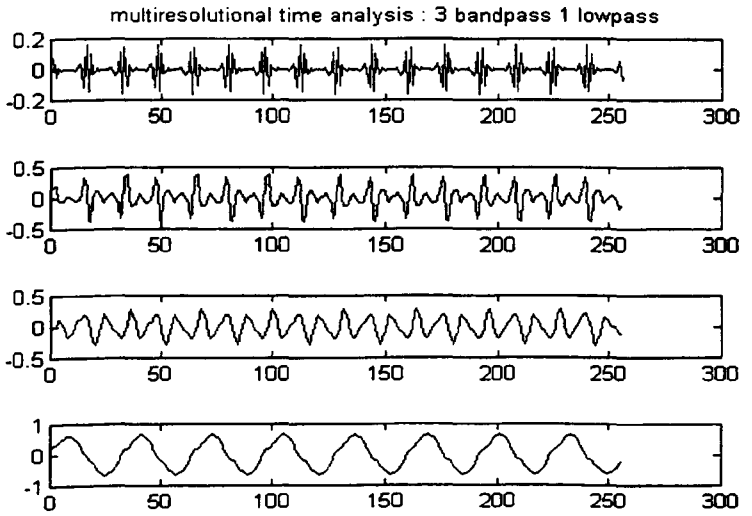


Fig.5.48. Daubechies8 analysis at 4 levels of test function 4. Remark at level 1 that only the transients of the square wave are shown. Level 4 shows a low pass filtering of the square wave : Only the fundamental wave (sine) is left.

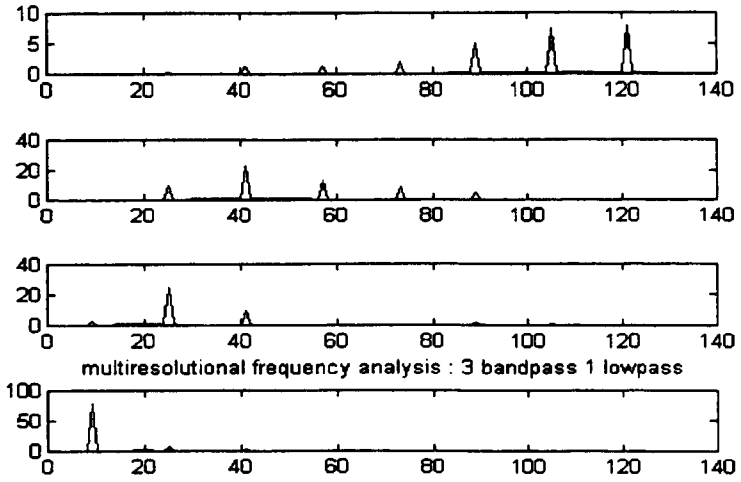


Fig.5.49. Spectral analysis of Daubechies8 analysed test function 4. 3 band pass levels and 1 low pass.

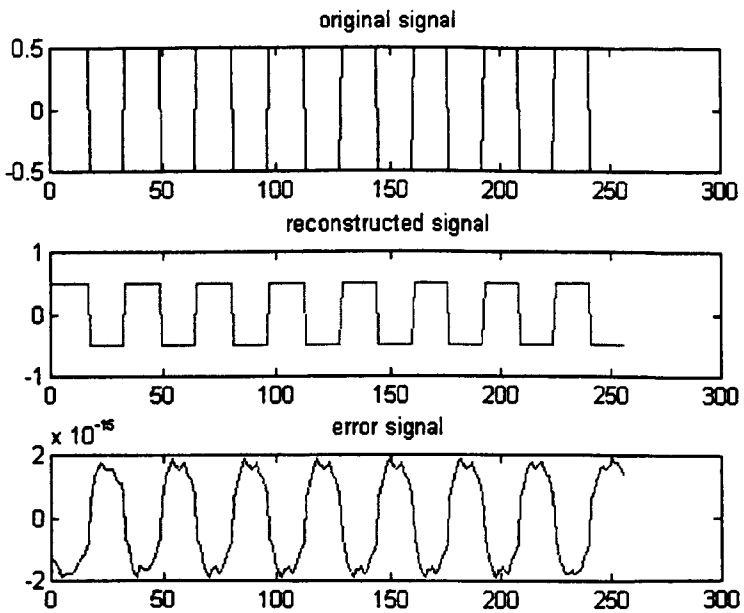


Fig.5.50. Error signal of Daubechies8 forwards and backwards transformed test function 4.

5.4.11. Symmetry

Compactly supported orthogonal wavelet bases all have a asymmetrical structure. This in contrast with the infinitely supported wavelet bases like for instance the Meyer one. In many applications, asymmetry can be a nuisance. In image coding for instance we are confronted with our visual system which is very sensible to phase linearity. Phase linear filters have symmetrical impulse responses. Wavelets with symmetrical filters coefficients are studied in the next chapter where we consider the new generation of wavelets among which are the **biorthogonal** wavelets.

CHAPTER 6

2D and Second Generation Wavelets.

6.1 Introduction

In Chapter 5 the evolution from classical Fourier to 1D wavelet transform was investigated. We now explore some new application fields like for instance the **lifting scheme** for biorthogonal wavelets. This leads us to the use of splines; the start of a new generation of wavelets which do not use the classical dilation-translation techniques anymore. Although not investigated here, they are a guide to, for instance non-uniform sampling and wavelets on spheres.

In this Chapter we consider 3 items :

- **2D wavelet algorithms**, as an extension of the 1D case, will be studied. In chapter 7 the algorithms will be applied in some image compression and noise reduction items.

- **Biorthogonal wavelets** can be seen as a first attempt to relax the stringent conditions put on orthogonal wavelets. Daubechies proved that asymmetry is a necessary condition for compactly supported orthogonal wavelets when analysis and synthesis coefficients are the same. By accepting complementary and not necessarily equal length sets for analysis and synthesis, symmetrical spline functions can for instance be considered to construct new kind of wavelets. This not only broadens the field it also opens new opportunities for faster and more efficient algorithms. In this context we will consider the biorthogonal splines .

- **The lifting scheme** is a new technique for implementing wavelets. It does not utilise the convolution techniques anymore but make use of 'predict' and 'update' techniques to construct and implement biorthogonal wavelets based on splines.

6.2. The Two Dimensional Wavelet Transform

For two dimensional signals, e.g., images, the transform consists in double application of the one dimensional DWT , once for the rows and a second time for the columns of an image. When using a scale factor two in both directions (commonly used) , it will be called the dyadic two dimensional DWT.

The splitting of the wavelet transform in two individual directions is a consequence of the fact that the transform is separable, because the basis functions $\phi(x_1)$, $\phi(x_2)$, $\psi(x_1)$ and $\psi(x_2)$ are orthonormal.

In figure 6.1. we can see how the data of the image is split up after one stage of the 2D WT . One such small square will be called a subband.

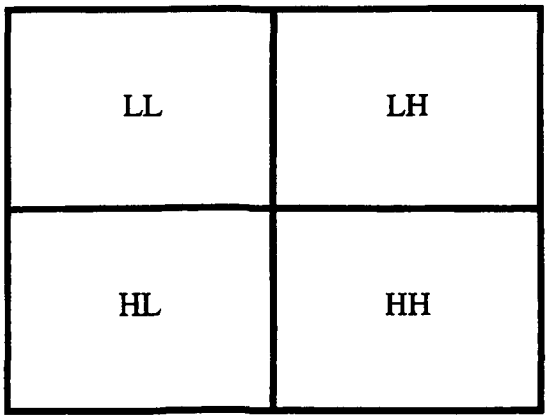


Figure 6.1. First stage of a 2-dimensional wavelet decomposition

The subbands contain:

- LL : all the low pass information : The result of decimation/convolution with the low pass filters h_1 and h_2.
- HH : all the high pass information : The result of convolution /decimation operation with the high pass filters g_1 and g_2.
- LH : horizontal high pass and vertical low pass information : the result of convolution /decimation operation with the high pass filter g_1 and the low pass filter h_2.
- HL : horizontal low pass and vertical high pass information : The result of convolution /decimation operation with the low pass filter h_1 and the high pass filter g_2.

In subsequent stages only LL is further transformed in a wavelet decomposition There is another alternative in which all 4 subbands are transformed in a wavelet packets decomposition. The practical wavelet decomposition and reconstruction of an image is illustrated in Fig.6.2. $\downarrow$ stands for down sampling (decimation) by 2; $\uparrow$ stands for up sampling (interpolation) by 2.

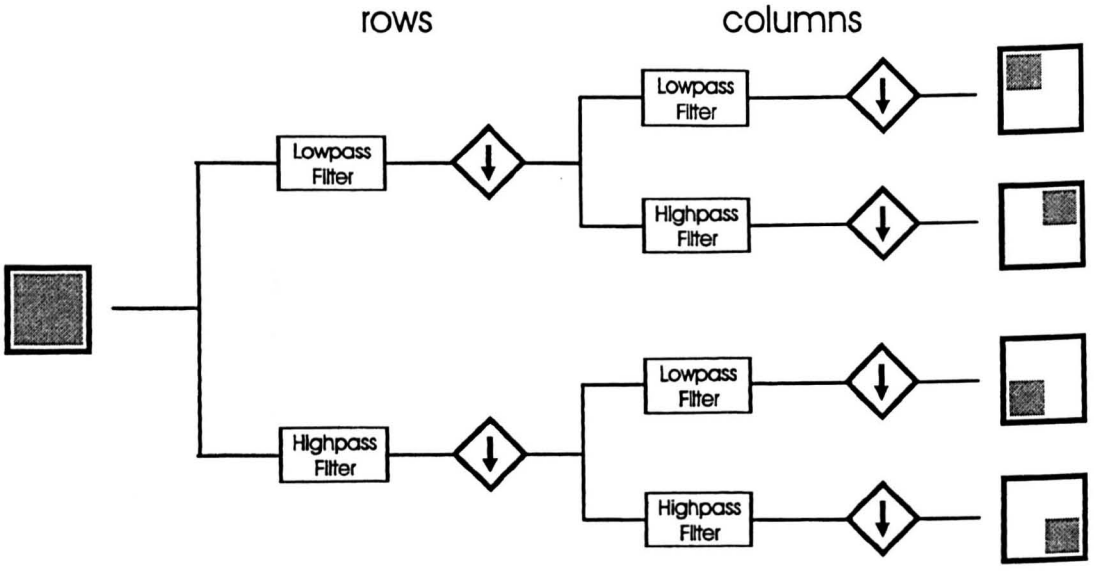


Figure 6.2. Two-dimensional wavelet decomposition.

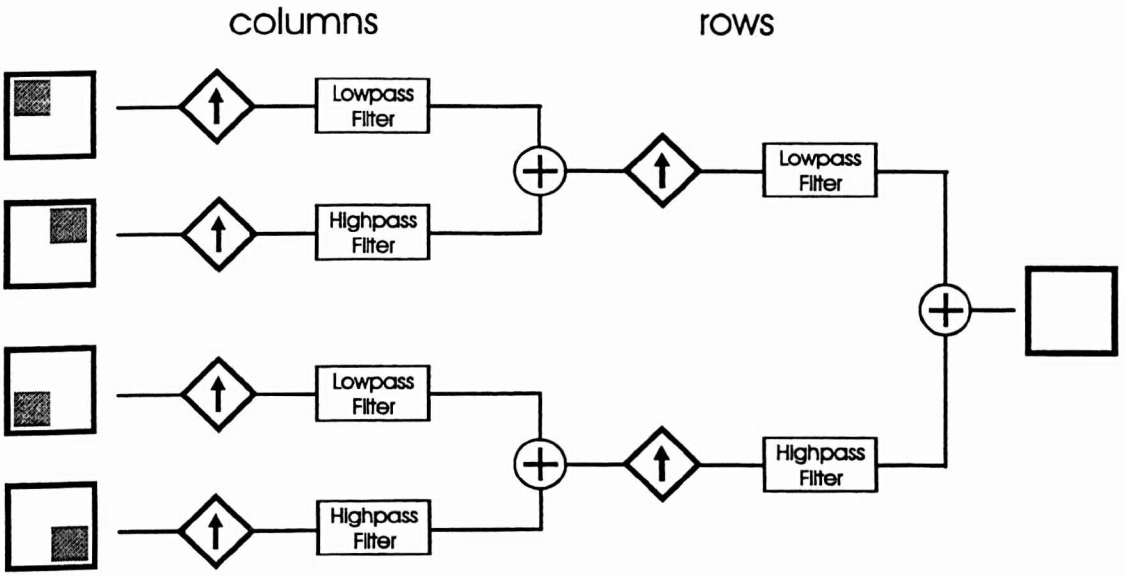


Figure 6.3. Two-dimensional wavelet reconstruction

The following part of the text shows a Mathcad program for 2D wavelet analysis and synthesis. Each part consists of 2 functions : the wavelet transform and the use of the wavelet transform at different levels. Haar and Daub4 are the easiest to implement and take the smallest amount of CPU time. This programming was intended to investigate Mallet's schema for 2D wavelet implementation. Chapter 7 investigates these algorithms on a video processor like the TMS320C80.

Note how, because of the separability of the 2D wavelet into 1D wavelets, the 1D transform can easily be implemented on the rows and the columns to result in a 2D decomposition and reconstruction. In Chapter 4 this algorithm was implemented in a first experiment to remove noise from a picture using Wiener filter techniques in a wavelet space.

2D WAVELET ANALYSIS

The Haar and Daubechies4 are available for 2D wavelet analysis. The analysis is composed out of two functions. One level analysis function called **wav_anal_level(f,wavelet)** and a global analysis function **wavelet_anal(f,wavelet,w_level).**

$$a := \frac{\sqrt{2}}{2} \qquad Haar\_low := \begin{pmatrix} a \\ a \end{pmatrix} \qquad Haar\_high := \begin{pmatrix} a \\ -a \end{pmatrix}$$

$$Haar := augment(Haar\_low, Haar\_high)$$

$$Daub4\_low := \begin{bmatrix} \frac{1+\sqrt{3}}{(4\cdot\sqrt{2})} \\ \frac{3+\sqrt{3}}{(4\cdot\sqrt{2})} \\ \frac{3-\sqrt{3}}{(4\cdot\sqrt{2})} \\ \frac{1-\sqrt{3}}{(4\cdot\sqrt{2})} \end{bmatrix} \qquad Daub4\_high := \begin{bmatrix} \frac{1-\sqrt{3}}{(4\cdot\sqrt{2})} \\ -1\cdot\frac{3-\sqrt{3}}{4\cdot\sqrt{2}} \\ \frac{3+\sqrt{3}}{(4\cdot\sqrt{2})} \\ -1\cdot\frac{1+\sqrt{3}}{4\cdot\sqrt{2}} \end{bmatrix}$$

$$Daub4 := augment(Daub4\_low, Daub4\_high)$$

$$
\begin{aligned}
&\text{wav\_anal\_level}(f, \text{wavelet}) := \\
&\quad \text{rows\_f} \leftarrow \text{rows}(f) \\
&\quad \text{trans\_f} \leftarrow f^{T} \\
&\quad \text{for } i \in 0..\text{rows\_f} - 1 \\
&\quad\quad f10^{\langle i \rangle} \leftarrow \text{response}\left(\text{trans\_f}^{\langle i \rangle}, \text{wavelet}^{\langle 0 \rangle}, \text{rows\_f}\right) \\
&\quad\quad f11^{\langle i \rangle} \leftarrow \text{response}\left(\text{trans\_f}^{\langle i \rangle}, \text{wavelet}^{\langle 1 \rangle}, \text{rows\_f}\right) \\
&\quad \text{for } i1 \in 0..\frac{\text{rows\_f} - 1}{2} \\
&\quad\quad f20^{\langle i1 \rangle} \leftarrow \left(f10^{T}\right)^{\langle 2 \cdot i1 + 1 \rangle} \\
&\quad\quad f21^{\langle i1 \rangle} \leftarrow \left(f11^{T}\right)^{\langle 2 \cdot i1 + 1 \rangle} \\
&\quad \text{for } i1 \in 0..\frac{\text{rows\_f} - 1}{2} \\
&\quad\quad f30^{\langle i1 \rangle} \leftarrow \text{response}\left(f20^{\langle i1 \rangle}, \text{wavelet}^{\langle 0 \rangle}, \text{rows\_f}\right) \\
&\quad\quad f31^{\langle i1 \rangle} \leftarrow \text{response}\left(f20^{\langle i1 \rangle}, \text{wavelet}^{\langle 1 \rangle}, \text{rows\_f}\right) \\
&\quad\quad f32^{\langle i1 \rangle} \leftarrow \text{response}\left(f21^{\langle i1 \rangle}, \text{wavelet}^{\langle 0 \rangle}, \text{rows\_f}\right) \\
&\quad\quad f33^{\langle i1 \rangle} \leftarrow \text{response}\left(f21^{\langle i1 \rangle}, \text{wavelet}^{\langle 1 \rangle}, \text{rows\_f}\right) \\
&\quad f30\_T \leftarrow f30^{T} \\
&\quad f31\_T \leftarrow f31^{T} \\
&\quad f32\_T \leftarrow f32^{T} \\
&\quad f33\_T \leftarrow f33^{T} \\
&\quad \text{for } i1 \in 0..\frac{\text{rows\_f} - 1}{2} \\
&\quad\quad f40^{\langle i1 \rangle} \leftarrow f30\_T^{\langle 2 \cdot i1 + 1 \rangle} \\
&\quad\quad f41^{\langle i1 \rangle} \leftarrow f31\_T^{\langle 2 \cdot i1 + 1 \rangle} \\
&\quad\quad f42^{\langle i1 \rangle} \leftarrow f32\_T^{\langle 2 \cdot i1 + 1 \rangle} \\
&\quad\quad f43^{\langle i1 \rangle} \leftarrow f33\_T^{\langle 2 \cdot i1 + 1 \rangle} \\
&\quad \text{f\_aug1} \leftarrow \text{augment}\left(f40^{T}, f41^{T}\right) \\
&\quad \text{f\_aug2} \leftarrow \text{augment}\left(f42^{T}, f43^{T}\right) \\
&\quad \text{stack}(\text{f\_aug1}, \text{f\_aug2})
\end{aligned}
$$

```
wavelet_anal(f, wavelet, w_level) := | col_f ← cols(f)
                                     | wav_lev ← wav_anal_level(f, wavelet)  if  w_level=1
                                     | otherwise
                                     |     wav_lev ← wav_anal_level(f, wavelet)
                                     |     f2 ← wav_lev
                                     |     count ← 1
                                     |     while count < w_level
                                     |         length_lev ← col_f / 2^count
                                     |         f1 ← submatrix(f2, 0, length_lev - 1, 0, length_lev - 1)
                                     |         f2 ← wav_anal_level(f1, wavelet)
                                     |         for i ∈ 0..length_lev - 1
                                     |           for j ∈ 0..length_lev - 1
                                     |             wav_lev_{i,j} ← f2_{i,j}
                                     |         count ← count + 1
                                     | wav_lev
```

2D WAVELET SYNTHESIS

The Haar and Daubechies 4 are available for 2D wavelet synthesis.Te inverse coefficients are used. The synthesis is composed of two functions.One level synthesis function called **wav_synt_level(f,rev_wavelet)** and the global function **wavelet _synt(f,rev_wavelet,w_level).**

$$a := \frac{\sqrt{2}}{2} \qquad Haar\_low := \begin{pmatrix} a \\ a \end{pmatrix} \qquad Haar\_high := \begin{pmatrix} a \\ -a \end{pmatrix}$$

$$Haar\_rev := augment(reverse(Haar\_low), reverse(Haar\_high))$$

$$Daub4\_low := \begin{bmatrix} \frac{1+\sqrt{3}}{(4\cdot\sqrt{2})} \\ \frac{3+\sqrt{3}}{(4\cdot\sqrt{2})} \\ \frac{3-\sqrt{3}}{(4\cdot\sqrt{2})} \\ \frac{1-\sqrt{3}}{(4\cdot\sqrt{2})} \end{bmatrix} \qquad Daub4\_high := \begin{bmatrix} \frac{1-\sqrt{3}}{(4\cdot\sqrt{2})} \\ -1\cdot\frac{3-\sqrt{3}}{4\cdot\sqrt{2}} \\ \frac{3+\sqrt{3}}{(4\cdot\sqrt{2})} \\ -1\cdot\frac{1+\sqrt{3}}{4\cdot\sqrt{2}} \end{bmatrix}$$

$$Daub4\_rev := augment(reverse(Daub4\_low), reverse(Daub4\_high))$$

```
wav_synt_level(f, rev_wavelet) :=
    row_f ← rows(f) / 2
    g0 ← submatrix(f, 0, row_f − 1, 0, row_f − 1)^T
    g1 ← submatrix(f, 0, row_f − 1, row_f, 2·row_f − 1)^T
    g2 ← submatrix(f, row_f, 2·row_f − 1, 0, row_f − 1)^T
    g3 ← submatrix(f, row_f, 2·row_f − 1, row_f, 2·row_f − 1)^T
    for i ∈ 0..row_f − 1
      for j ∈ 0..2·row_f − 1
        zero1_{i,j} ← 0
    zero2 ← stack(zero1, zero1)
    g40 ← zero1
    g41 ← zero1
    g42 ← zero1
    g43 ← zero1
    for j1 ∈ 0..row_f − 1
      g40^<2·j1> ← g0^<j1>
      g41^<2·j1> ← g1^<j1>
      g42^<2·j1> ← g2^<j1>
      g43^<2·j1> ← g3^<j1>
    g40_T ← g40^T
    g41_T ← g41^T
    g42_T ← g42^T
    g43_T ← g43^T
    for j1 ∈ 0..row_f − 1
      g30^<j1> ← response(g40_T^<j1>, rev_wavelet^<0>, 2·row_f)
      g31^<j1> ← response(g41_T^<j1>, rev_wavelet^<1>, 2·row_f)
      g32^<j1> ← response(g42_T^<j1>, rev_wavelet^<0>, 2·row_f)
      g33^<j1> ← response(g43_T^<j1>, rev_wavelet^<1>, 2·row_f)
    g20 ← g30 + g31
    g21 ← g32 + g33
    g10 ← zero2
    g11 ← zero2
    for j1 ∈ 0..row_f − 1
      g10^<2·j1> ← g20^<j1>
      g11^<2·j1> ← g21^<j1>
    g10_T ← g10^T
    g11_T ← g11^T
    for j2 ∈ 0..2·row_f − 1
      g0^<j2> ← response(g10_T^<j2>, rev_wavelet^<0>, 2·row_f)
      g1^<j2> ← response(g11_T^<j2>, rev_wavelet^<1>, 2·row_f)
    g ← g0^T + g1^T
```

```
wavelet_synt (f, rev_wavelet, w_level) := | col_f ← cols(f)
                                          | wav_lev ← wav_synt_level (f, rev_wavelet) if w_level = 1
                                          | otherwise
                                          |   wav_lev ← f
                                          |   count ← w_level - 1
                                          |   while count ≥ 0
                                          |     length_lev ← col_f / 2^count
                                          |     f1 ← submatrix(wav_lev, 0, length_lev - 1, 0, length_lev - 1)
                                          |     f2 ← wav_synt_level (f1, rev_wavelet)
                                          |     for i ∈ 0..length_lev - 1
                                          |       for j ∈ 0..length_lev - 1
                                          |         wav_lev_{i,j} ← f2_{i,j}
                                          |     count ← count - 1
                                          | wav_lev
```

6.3 Biorthogonal Wavelets

The orthogonality property causes strong limitations for the construction of wavelets. The Haar is the only real-valued wavelet, using the same scaling and wavelet function for analysis and synthesis, that is o compactly supported, symmetric and orthogonal. All the others are non-symmetric have non-linear phase and thus producing distortion while processing images .

Constraints has to be relaxed for instance on the uniqueness of the basis functions for forward and backward WT. **Dual scaling function** $\tilde{\varphi}$ and **dual wavelets** $\tilde{\psi}$ are to be considered .They generate a dual multiresolution analysis with subspaces $\tilde{V}_j$ and $\tilde{W}_j$, so that :

$$\tilde{V}_i \perp W_i \quad \text{and} \quad V_i \perp \tilde{W}_i \tag{6.1}$$

and, consequently,

$$\tilde{W}_j \perp W_{j'} \quad \text{for} \quad j \neq j'. \tag{6.2}$$

The dual basis is constructed with **dual wavelets** $\tilde{\psi}_{j,k}$ which are orthogonal to the wavelets $\psi_{j,k}$,

$$\left\langle \psi_{j,k}, \tilde{\psi}_{j',k'} \right\rangle = \delta_{k,k'}\delta_{j,j'} \tag{6.3}$$

$\{h, \tilde{h}, g, \tilde{g}\}$ is a set of **biorthogonal filters.** $\tilde{h}$ and $\tilde{g}$ are defined by

$$\tilde{h}_{k-2m} = \left\langle \tilde{\varphi}(t-m), \varphi(2t-k) \right\rangle$$

$$\tilde{g}_{k-2m} = \left\langle \tilde{\psi}(t-m), \varphi(2t-k) \right\rangle \tag{6.4}$$

The resulting scaling functions, wavelets, dual scaling functions and dual wavelets are biorthogonal in the way that :

$$\begin{aligned} \left\langle \tilde{\varphi}_{j,k}, \varphi_{j,k'} \right\rangle &= \delta_{k,k'} \\ \left\langle \tilde{\psi}_{j,k}, \psi_{j,k'} \right\rangle &= \delta_{k,k'} \\ \left\langle \tilde{\varphi}_{j,k}, \psi_{j,k} \right\rangle &= 0 \\ \left\langle \tilde{\psi}_{j,k}, \varphi_{j,k} \right\rangle &= 0. \end{aligned} \tag{6.5}$$

The dual multiresolution analysis is not necessarily the same as the one generated by the original basis functions. In signal processing terms this means that the filters used in the forward transform are different from those in the inverse transform. However, the inverse filters are symmetric variants of the forward filters.

Experimentally we found out (Appendix 5a) but Daubechies also proves [57] that filters for two-band splitting schemes, like the wavelet filters h and g, cannot be orthogonal and linear phase simultaneously when the filters have a length greater than 2. The Haar filters are the only ones that comply. Though, linear phase can be introduced in the filters by accepting biorthogonality.

In sound coding, linear phase is not important, because the human hearing system is not phase-sensitive. This is however not the case for images. Thus, 2D filters for images should have neither amplitude distortion, nor phase distortion.

The idea of multiresolution can be extended to describe the biorthogonal case as well. Biorthogonal filters are different for decomposition and reconstruction, as seen

in (Fig. 6.4.). This is because the orthogonality requirement is replaced by the weaker requirement of biorthogonality. Mostly, the number of filter coefficients are not the same. They can be designed with the same techniques in mind as with the Daubechies wavelets :

- Put the biorthogonal forward and inverse transform in a matrix form and deduce some equation for orthogonality out of it.
- Formulate some extra constraints of vanishing moments.
- Construct a set of non-linear equations and solve. (Appendix 6a)

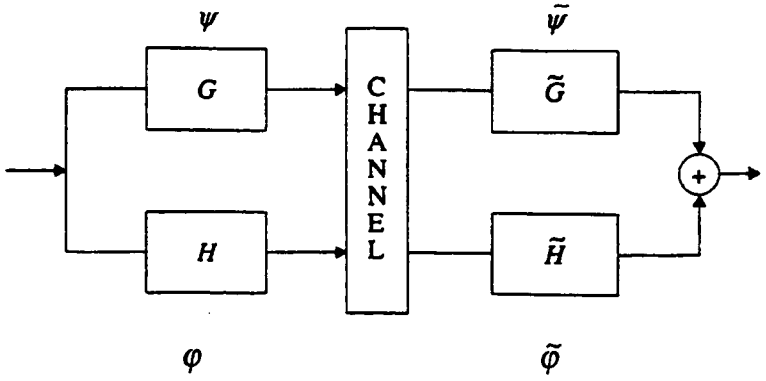


Figure 6.4. Schematic filter-bank scheme for the biorthogonal wavelet transform. Mallat's scheme can be used unmodified for different levels. Only the filter coefficients will change in the reconstruction phase.

In the following table a simple comparison is made between former transformations and the biorthogonal one. Note, that the Gabor transform, explained in Chapter 4, had no variable resolution windows.

Property	Gabor	Normal Wavelet	Biorthogonal Wavelet
Reconstruction	not perfect	perfect	perfect
Basis functions	not orthogonal	orthogonal	biorthogonal
Phase	linear	non-linear	linear
Filter length	relatively long	relatively short	relatively short

Table 6.1 : Comparison of some properties of transformations

Figure 6.5. shows the biorthogonal cdf17-15 (Cohen, Daubechies & Feauveau) filter, with:

(a) : The magnitude-frequency response

(b) : The magnitude-frequency response in dB

(c) : The phase-frequency response

(d, e) : The scaling functions

(f, g) : The wavelet functions

Note in Fig. 6.5.a. the 'non-mirror' shape of the frequency responses and in Fig.c the linear phase response. Note also that the figures d-g form the scaling wavelet set for analysis and that the figures e-f form the set for synthesis

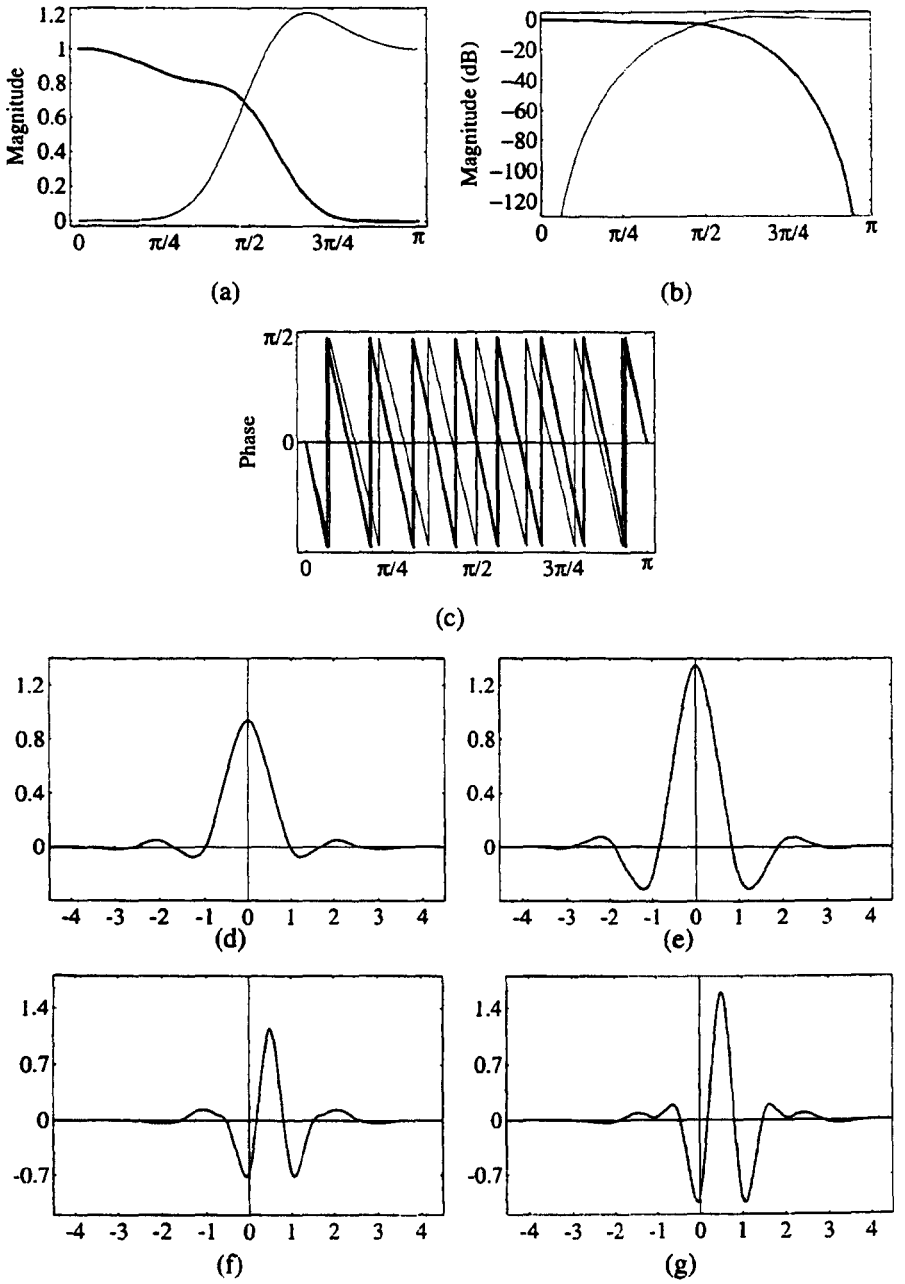


Figure 6.5. Cdf17-15 characteristics

6.4. The Lifting Scheme

6.4.1. Introduction

The **lifting scheme** is a new method for constructing biorthogonal wavelets. The construction takes place entirely in the **spatial domain** since it does not rely on convolution and Fourier theory as the classical WT algorithms do.

There are three major steps in the construction of wavelets using lifting. The first step or **Lazy wavelet** splits the samples into even and odd subsets. The second one calculates the high pass wavelet coefficients (γ) as the failure to **predict** the odd set based upon the even set. The third step **updates** the odd set using **scaling functions** formed by the calculated high pass wavelet coefficients in order to find the low frequency values (λ coefficients). As in classical wavelets we want to come to the expression for a general function f

$$f = \sum_i \gamma_i \psi_i$$

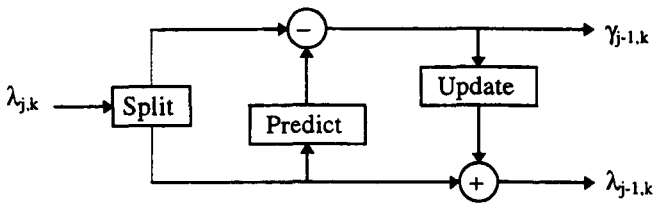


Fig.6.6. The lifting scheme.

Wim Sweldens et al. [11] are very active in this research area. In this chapter we will use Sweldens' notation (λ and γ) for the respectively low pass and high pass filtered wavelet coefficients. Because of this new approach to wavelet design and the opportunity to head for new application fields we will call them **2nd generation wavelets**.

The basic idea behind the name **lifting scheme** is to start with a trivial wavelet, the Lazy wavelet which essentially doesn't calculate anything of the formal

properties of a wavelet. Although, by splitting the even from the odd samples it performs a first step in the direction of **decorrelating** the signal in a **broad sense**. The lifting scheme which comes after the Lazy decomposition, gradually builds a new wavelet with improved properties.

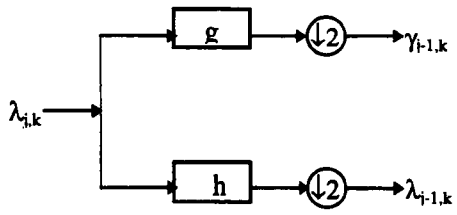


Figure 6.7. Classical fast wavelet transform (Mallat's algorithm).

Figure 6.7. shows the classical fast wavelet transform. In this algorithm, the input is filtered by both the high pass filter g as the low pass filter h and the two results are subsampled.

In the fast lifted wavelet transform, low pass coefficients will be calculated with the help of earlier calculated high pass coefficients. This will avoid subsampling of the results. We will also prove that, by using the lifting scheme, the amount of FLOPs can be reduced by a factor of two.

6.4.2. Split Step

If we have a sampled signal, we denote the original samples as $\lambda_{0,k} = f(k)$ for $k \in Z$. The number of coefficients can now be reduced by subsampling the even samples of the original signal : $\lambda_{0,2k}$.

$$\lambda_{-1,k} = \lambda_{0,2k} \quad \text{for } k \in Z \tag{6.6}$$

The wavelet coefficients $\gamma_{-1,k}$ are used to encode the lost information, so we can reconstruct the original set $\lambda_{0,k}$ from the set $\lambda_{-1,k}$. In the case of the Lazy wavelet we say that the lost information is contained in the odd coefficients. Of course, this does not ideally decorrelate the signal.

$$\gamma_{-1,k} = \lambda_{0,2k+1} \quad \text{for } k \in Z \tag{6.7}$$

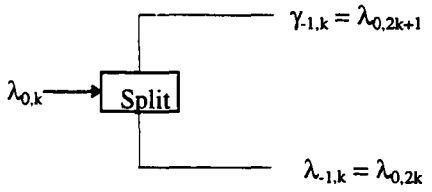


Figure6.8. Split step.

Note that there is no restriction on the way how the data should be split, nor on the size of the subsets. Splitting between even and odd points in the Lazy wavelet is a good choice to start with. Remember that we are looking for a decorrelation and because the correlation between neighbouring points is generally good, splitting them up into an even and an odd set is a first primitive step towards decorrelation.

6.4.3. Predict Step

The Lazy wavelet only subsamples the data in even and odd samples and will not help to efficiently decorrelate the signal.

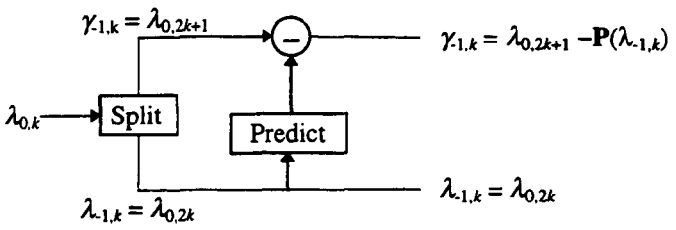


Figure 6.9. Split and predict step.

The odd samples $\lambda_{0,2k+1}$ can be predicted on basis of the even samples $\lambda_{-1,k}$ found by the Lazy wavelet. For example an odd sample $\lambda_{0,2k+1}$ can be predicted as the average of it's two even neighbours $\lambda_{-1,k}$ and $\lambda_{-1,k+1}$ (Fig. 6.10). This will not be altogether exact but it will be close to the original odd sample if there is correlation between subsequent samples in the signal. As we are interested in the 'detail' in a signal we define the wavelet coefficient as the difference between the original sample and the predicted value.

$$\gamma_{-1,k} = \lambda_{0,2k+1} - \mathbf{P}(\lambda_{-1,k}) \qquad (\ \mathbf{P}: \text{Predict}\ )$$

$$\gamma_{-1,k} = \lambda_{0,2k+1} - \tfrac{1}{2}(\lambda_{-1,k} + \lambda_{-1,k+1}) \qquad (6.8)$$

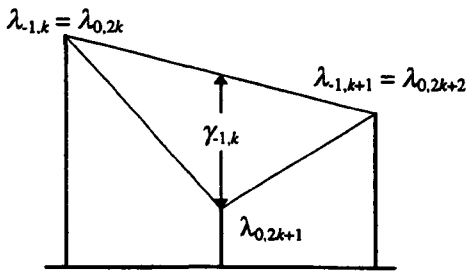


Figure 6.10. Wavelet coefficients as the difference between sample value and prediction (linear interpolation case)

If there exists correlation between subsequent samples, most of the wavelet coefficients will be rather small. If the signal is piecewise linear between the even samples, all wavelet coefficients are zero. The **wavelet coefficients** are in fact a **measure of the strength of the signal's non-linearity**. In terms of frequency content, they capture the higher harmonics of the signal in fact the high frequencies while the $\lambda_{-1,k}$ capture the low frequencies.

Instead of using a linear prediction, the failure to be quadratic, cubic (or an higher order) can be computed. This leads to the concept of **interpolating subdivision** which is discussed further.

6.4.4. Update Step

The earlier mentioned steps of splitting and prediction can now be iterated until the original data is replaced with the wavelet representation. However, the choice of $\lambda_{-1,k}$ with Lazy is not the best one. These coefficients are found by a simply subsampling of the original signal, so their frequency content stretches out over the whole band of the original signal. By subsampling the signal, the sampling frequency is halved hence aliasing is possible when for instance the frequency components of the $\lambda_{-1,k}$ exceed half the new sampling frequency .

If there are 2^n samples, the split and prediction steps can be applied n times resulting in the coefficients $\lambda_{j,k}$ ($-n \leq j \leq -1$, $0 \leq k \leq 2^n-1$) and two remaining (low pass) coefficients $\lambda_{-n,0}$ and $\lambda_{-n,1}$. These two coefficients are respectively the first ($\lambda_{0,0}$) and the last ($\lambda_{0,2^n-1}$) of the original samples.

It is desirable to keep some of the global properties of the original data set in the new higher level set : For instance that lower scale images should have the same brightness or average pixel value as higher scale ones. This implies that the last value should be the average of all pixel values in the original image. This last value is always the low pass remaining λ coefficient. Since the high pass wavelet coefficients $\gamma_{j,k}$ always have an average equals to zero there is only the DC value left for the last 'approximation' coefficient..

The average value of the$\lambda_{j,k}$ should be the same for each scale. With the decrease by 2 of the amount of samples we can write :

$$\sum_k \lambda_{j-1,k} = \tfrac{1}{2} \sum_k \lambda_{j,k} \qquad (6.9)$$

The low pass coefficients $\lambda_{-1,k}$ will be lifted with the calculated neighbouring wavelet coefficients $\gamma_{j-1,k}$:

$$\lambda_{j-1,k} = \lambda_{j-1,k} + \mathbf{U}(\gamma_{j-1,k}) \qquad (\text{U : Update})$$

$$\lambda_{j-1,k} = \lambda_{j-1,k} + A(\gamma_{j-1,k-1} + \gamma_{j-1,k}) \qquad (6.10)$$

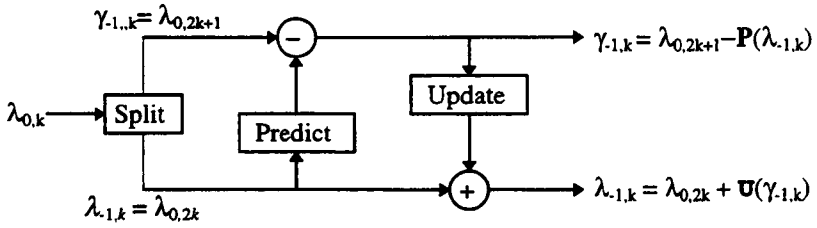


Figure 6.11. Split, predict and update step for the first level.

A can be found by calculating the average :

$$\sum_k \lambda_{j-1,k} = \sum_k \lambda_{j,2k} + 2A\sum_k \gamma_{j-1,k} = (1-2A)\sum_k \lambda_{j,2k} + 2A\sum_k \lambda_{j,2k+1} \tag{6.11}$$

To maintain the average A = ¼ .Then (6.10) becomes :

$$\lambda_{j-1,k} := \lambda_{j-1,k} + ¼\,(\gamma_{j-1,k-1} + \gamma_{j-1,k}) \tag{6.12}$$

6.4.5. One Level in the Wavelet Decomposition

The split, predict and update steps can be iterated until the original data is replaced by wavelet coefficients $\gamma_{j,k}$ and one low pass coefficient $\lambda_{-n,0}$. One level of the wavelet decomposition is depicted in figure 6.12.

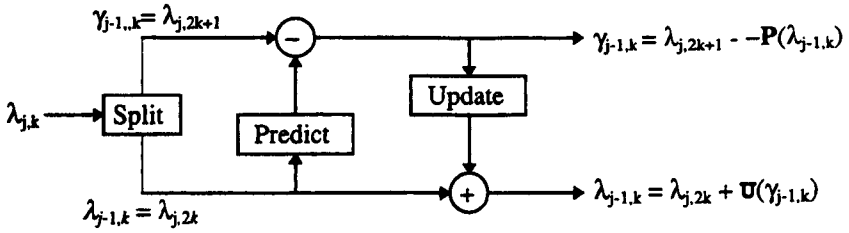


Figure 6.12 One level in the wavelet decomposition with use of the lifting scheme.

The split, predict and update steps can be combined into two steps. When the notation of figure 6.15 is used, one level of the wavelet decomposition of the above explained example is given by Algorithm 6.1

Each level of the wavelet transform consists of two stages :

- calculate wavelet coefficients as the failure to be linear, from (6.8)

$$\gamma_{j-1,k} := \lambda_{j,2k+1} - \tfrac{1}{2}(\lambda_{j,2k} + \lambda_{j,2k+2})$$

- lift the subsampled coefficients (6.7) with these wavelet coefficients, from (6.12)

$$\lambda_{j-1,k} := \lambda_{j,2k} + \tfrac{1}{4}(\gamma_{j-1,k-1} + \gamma_{j-1,k})$$

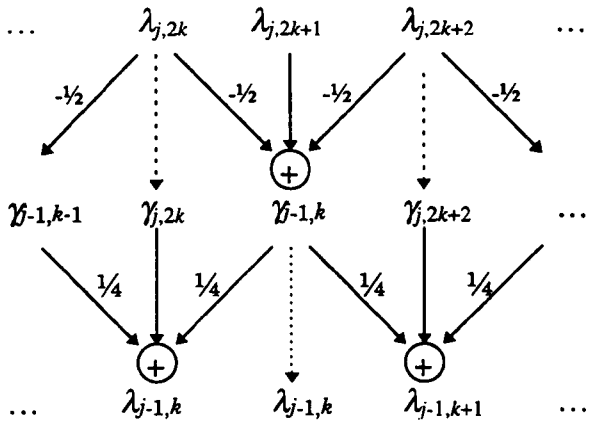


Figure 6.13. Lifting scheme with prediction and update (1 scale)

The inverse transform simply undoes the steps, done in the forward transform:

- undo the lifting

$$\lambda_{j,2k} := \lambda_{j-1,k} - \tfrac{1}{4}(\gamma_{j-1,k-1} + \gamma_{j-1,k})$$

- restore the data from the prediction

$$\lambda_{j,2k+1} := \gamma_{j-1,k} + \tfrac{1}{2}(\lambda_{j,2k} + \lambda_{j,2k+2})$$

Algorithm 6.1 : One scale in the lifted wavelet decomposition and reconstruction applied to an example.

6.4.6. Other Prediction Functions : Interpolating Subdivision

As mentioned before, other (non-linear) prediction methods are possible. This leads us to the **interpolation subdivision**. We use the value N+1 to denote the order of the subdivision scheme. The piecewise linear approximation as seen in the earlier section, uses N = 1. For a cubic interpolation N should be three. N determines the smoothness of the interpolation function used in the lifting scheme process to find the wavelet coefficients .

Cubic interpolation is now explained by an example. When two neighbouring points on either side of a central point are used to predict it, a cubic polynomial p(t) can be designed which interpolates those four values

$$\begin{aligned} \lambda_{j,k-1} &= p(t_{j,k-1}) \\ \lambda_{j,k} &= p(t_{j,k}) \\ \lambda_{j,k+1} &= p(t_{j,k+1}) \\ \lambda_{j,k+2} &= p(t_{j,k+2}) \end{aligned} \tag{6.13}$$

The predicted sample value (even sample with odd index) will be the value that the polynomial takes on at the midpoint, leaving the odd samples (with even index) preserved

$$\begin{aligned} \lambda_{j+1,2k+1} &= p(x_{j+1,2k+1}) \\ \lambda_{j+1,2k} &= \lambda_{j,k} \end{aligned} \tag{6.14}$$

Fig. 6.14 shows one iteration step in the process of cubic interpolation. Considering the fact that all the values are coming from piecewise polynomials, the limit function, resulting from an infinite interpolation process could be of particular interest. Remember the scaling function and the wavelet function which resulted from an interpolation process (in fact the inverse wavelet transform; see Fig.5.35,36,37) The limit function is not a polynomial anymore but Unser [20,21] showed that spline polynomials result in very useful new scaling functions. Orthogonality is a too coercive demand but if biorthogonality is acceptable one can come to very simple wavelet filter coefficients. The basic splines $\beta^N(t)$ for example are constructed by a N+1 times repeated convolution of a B-spline of order 0 , where $\beta^0(t)$ is the characteristic function in the interval [0,1) . The relation between successive scales is determined by

$$p\ (l) = \sum_{k=-\infty} n\ (K) p\ (2l - K)$$

where h^N is the binomial kernel of order N,

$$n\ (K) = \left\{ \frac{}{2^N} C_{k+N+1/2} \right\}, |K| \leq (N+1)/2 \quad \xleftrightarrow{DFT} \quad H^N(\Omega) = 2\cos^{N+1}(\Omega/2) \qquad (6.15)$$

$$= 0 \text{ otherwise}$$

Note that for N=0, this is the spectrum initially used in Appendix 5A which resulted in the 'rediscovery' the Haar wavelet. Broadening this design idea for N>0 and considering orthogonality result in a rational numbered sets of filter coefficients. (Table 6.3).

Another intriguing idea is the fact that by using these polynomials the **samples shouldn't necessary be located at integers**. Sweldens [12] describes in this context how **irregularly sampled wavelets** could be constructed.

Proceeding with the analysis and stating that N+1 samples are used and polynomials of degree N are build. We call N+1 the **order** of the subdivision scheme.

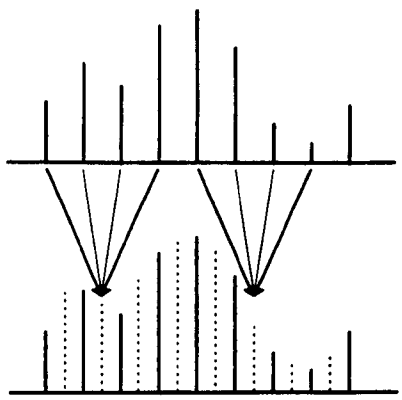


Figure 6.14. Interpolation subdivision with cubic polynomial.

The prediction function **P** now uses a polynomial interpolation of order N to find the predicted values. The higher the order of this function, the better the approximation of the γ coefficients based on the λ coefficients. This is very useful if

the original data evolves like some polynomial of order N. (Order of images : 2-3). Then, the predicted set of wavelet coefficients $\gamma_{j,k}$ tends to be zero because there is almost no difference between original and predicted values.

The interesting point of interpolation subdivision is that there is only one routine needed to construct an interpolating polynomial, given the sample values and their locations. The new sample value is then given by the evaluation of the polynomial at the new location. Neville's algorithm [55] is a suitable method for this purpose.

When finite sequences are processed, boundaries have to be taken into account. With N=4, there are four possible cases. Note that due to splitting there is always a λ in the first position.

- Near left boundary : more λ coefficients on the right side of the γ coefficient
  - 0 λ's on the left, 4 λ's on the right (does never occur)
  - 1 λ on the left, 3 λ's on the right
- Middle : enough coefficients on either side of the γcoefficient
  - 2 λ's on the left, 2 λ's on the right
- Near right boundary : more λ coefficients on the left side of the γ coefficient
  - 3 λ's on the left, 1 λ on the right
  - 4 λ's on the left, 0 λ's on the right

With the interpolation scheme and Neville's algorithm, a set of coefficients is generated. They are used to find the correct approximation, using a function of order N. With N = 1 , there are two coefficients for the two possible cases (one λ on the left and one λ on the right or 2 λ's on the left and none on the right). For N = 3, there are the four cases mentioned above.

Results for the linear and the cubic polynomial interpolation can be found in the tables 6.1 and 6.2. Note that the first case never occurs because of the splitting method. Note also the symmetry.

Cases		Coefficients			
# λ's on the left	# λ's on the right	$k-3$	$k-1$	$k+1$	$k+3$
0	2			-½	$^3/_2$
1	1		½	½	
2	0	$^3/_2$	-½		

Table6.1 : Filter coefficients for N = 1 (linear subdivision)

Cases		Coefficients							
# λ's on the left	# λ's on the right	$k-7$	$k-5$	$k-3$	$k-1$	$k+1$	$k+3$	$k+5$	$k+7$
0	4					$^{35}/_{16}$	$-^{35}/_{16}$	$^{21}/_{16}$	$-^{5}/_{16}$
1	3				$^{5}/_{16}$	$^{15}/_{16}$	$-^{5}/_{16}$	$^{1}/_{16}$	
2	2			$-^{1}/_{16}$	$^{9}/_{16}$	$^{9}/_{16}$	$-^{1}/_{16}$		
3	1		$^{1}/_{16}$	$-^{5}/_{16}$	$^{15}/_{16}$	$^{5}/_{16}$			
4	0	$-^{5}/_{16}$	$^{21}/_{16}$	$-^{35}/_{16}$	$^{35}/_{16}$				

Table6.2. Filter coefficients for N = 3 (cubic subdivision).

6.4.7. Interpolating Subdivision in the Update Step

In the update step (6.6.4) the $\lambda_{-1,k}$ values are lifted with the help of the neighbouring wavelet coefficients. The goal is to find a new set of $\lambda_{-1,k}$ values with preserved mean . If necessary higher moments can also be considered. They would result in longer update algorithms . This would be the same as considering longer filter coefficients in the classical DWT.

The final updated coefficients can be calculated with a new operator extracting $\lambda_{-1,k}$ **directly** from $\lambda_{0,k}$. However, this creates a scheme that is hardly to invert and it o does not reuse the work already done. For this reason, we will rather use the previously calculated set of wavelet coefficients $\gamma_{-1,k}$ to update $\lambda_{-1,k}$ in such a way that, at least, the mean is preserved. This decision is incorporated in the already given formula (6.10)

$$\lambda_{j-1,k} = \lambda_{j-1,k} + U(\gamma_{j-1,k}) \tag{6.16}$$

The associated scaling function can now be found. The method was used in Chapter 5 and evolves as follows : set all $\lambda_{0,k}$ values to zero except for $\lambda_{0,0}$ which is set to one and run the interpolating subdivision **ad infinitum**

The resulting function is the scaling function $\varphi(t)$, which is used to create a real wavelet that will maintain some desired properties from the original signal. This function will have an order $\tilde{N}$, which is not necessarily equal to N.

Considering higher moments in wavelet design showed to be very useful in the design of wavelet filter coefficients (Appendix 5B). We requested

$$\int \psi(t)dt = 0, \quad \int t\psi(t)dt = 0, \quad \int t^2\psi(t)dt = 0, \quad \ldots, \quad \int t^N\psi(t)dt = 0, \tag{6.17}$$

$\tilde{N}$ moments of the λ values has to be preserved at every level . This information is then used to see how much of a γ coefficient is needed to update a λ. These update coefficients are named the **lifting coefficients.**

In [54] an algorithm is presented to calculate these lifting coefficients. In this algorithm one has to solve a linear system of $\tilde{N}$ x N variables. The algorithm delivers coefficients for the different possibilities at the boundaries. However, for signals with larger duration, the boundaries do not affect most of the λ coefficients. Then, if N = $\tilde{N}$ = *1*, the lifting coefficients are going to be (¼,¼) for every λ unaffected by the boundaries. This was already suggested in (6 .12)

$$\lambda_{j-1,k} = \lambda_{j-1,k} + \tfrac{1}{4}(\gamma_{j-1,k-1} + \gamma_{j-1,k}) \tag{6.18}$$

6.4.8. Fast Lifted Wavelet Transform

Forward Fast Lifted Wavelet transform

for j = -1 down to -n

$$\begin{cases} \{\lambda_j, \gamma\} := Split(\lambda_{j+1}) \\ \gamma_j \mathrel{-}= P(\gamma_j) \\ \lambda_j \mathrel{+}= U(\gamma_j). \end{cases}$$

Inverse Fast Lifted Wavelet Transform

for j = -n to -1

$$\begin{cases} \lambda_j \mathrel{-}= U(\gamma_j) \\ \gamma_j \mathrel{+}= P(\lambda_j) \\ \lambda_{j+1} := Join(\lambda_j, \gamma_j). \end{cases}$$

Algorithm6.2 : Forward and inverse fast Lifted wavelet transform.

6.4.10. Implementation of the Fast Lifted Wavelet Transform with B-Spline Filters

In the case of linear subdivision, the filter coefficients associated to $\beta^1(t)$ are :

$$h = \{ \tfrac{1}{2} \; 1 \; \tfrac{1}{2} \}.$$

The associated scaling function is the well known linear B-spline **hat** or **Franklin** function. In the case of cubic interpolation the filter coefficients are :

$$h = \{(0 \; -1 \quad 1 \quad 8 \quad 8 \quad 1 \; -1) / 16 \}.$$

The even numbered coefficients are $h_{2k} = \delta_{k,0}$ thus they are one only if the indexes mach the processed points (γ), otherwise they are zero.

In Table 6.3, some filters are depicted which originate from Cohen-Daubechies-Feauveau (CDF-filters).

Type	high pass filter coefficients (h)	low pass filter coefficients ($\tilde{h}$)
[1 1]	0 ½ ½	0 ½ ½
[1 3]	0 ½ ½	(0 -1 1 8 8 1 -1) / 16
[1 5]	0 ½ ½	(0 3 -3 -22 -22 128 128 22 -22 -3 3) / 256
[2 2]	¼ ½ ¼	-1/8 ¼ ¾ ¼ -1/8
[2 4]	¼ ½ ¼	(3 -6 -16 38 90 38 -16 -6 3) / 128
[2 6]	¼ ½ ¼	(-5 10 34 -78 -123 324 700 324 -123 -78 34 10 -5) / 1024
[3 1]	0 1/8 3/8 3/8 1/8	0 -¼ ¾ ¾ -¼
[3 2]	0 1/8 3/8 3/8 1/8	(0 3 -9 -7 45 45 -7 -9 3 0) / 64
[3 3]	0 1/8 3/8 3/8 1/8	(0 -5 15 19 -97 -26 350 350 -26 -97 19 15 5) / 1024

Table6.3. Classical B-spline filter and lifting coefficients.

Some of the associated filter and lifting coefficients used in the lifting scheme are

$[N\ \tilde{N}]$	filter coefficients	lifting coefficients
[2 2]	½ 1 ½	¼ 1 ¼
[2 4]	½ 1 ½	(-6 0 38 1 38 0 -6) / 128 or

Table 6.4. B-spline filter and lifting coefficients.

Table 6.4 should be interpreted as this : the coefficients with value 1 in the middle are the coefficients that need to be approximated (γ) or updated (λ). In those cases the zero-valued coefficients are the neighbouring values that need to be filtered or lifted. The relation between Tables 6.3 and 6.4 are as follows :

- the odd new filter coefficients are the double of the high pass filter coefficients
- the odd lifting coefficients are the same as the low pass filter coefficients
- the even filter or lifting coefficients are zero, except in "the middle".

Note that for certain filters all coefficients have values equal to the inverse of a power of two. This is interesting for implementation in the VSP (Video Signal Processor) where only bit shifts, thus powers of two, are allowed.

The two shaded types of filter and lifting coefficients are the most appropriate for our goal. For the second one, the lifting coefficients are not longer an inverse power of two but they are at least a multiple of powers of two.

It must be mentioned that Table 6.3 only applies for points not affected by boundaries; otherwise we have to calculate the values with a linear equation or use tables like 6.1 and 6.2.

6.4.11. Example and In-Place Calculation

The same example is used as in the classical calculation of the biorthogonal wavelet transform (appendix 6.1). B-spline biorthogonal filters are used with N $= \tilde{N} = 1$.

In the explanation of the interpolation subdivision, boundary effects were minimised by choosing more coefficients to filter or update on one side (tables 6.1 and 6.2). But to obtain exactly the same values as the classical biorthogonal wavelet coefficients computed by Wavelab ®, the algorithm has to be simplified. Instead of choosing more values at one side of the predicted or updated coefficient, coefficients at the other end are used. This is not so good as the interpolating subdivision solution because the correlation between those points will normally be minimal. For example, a pixel on the left side of an image will normally have no relation with one on the right side. However, when dealing with large datasets, the boundary impact is minimal. In real-time situations there are maybe even no boundaries.

1. The original samples $\lambda_{0,k}$
2. Then, the wavelet coefficients $\gamma_{i,k}$ are calculated with (6.8)
3. Finally, the odd samples (even index) are lifted with these coefficients using (6.12)

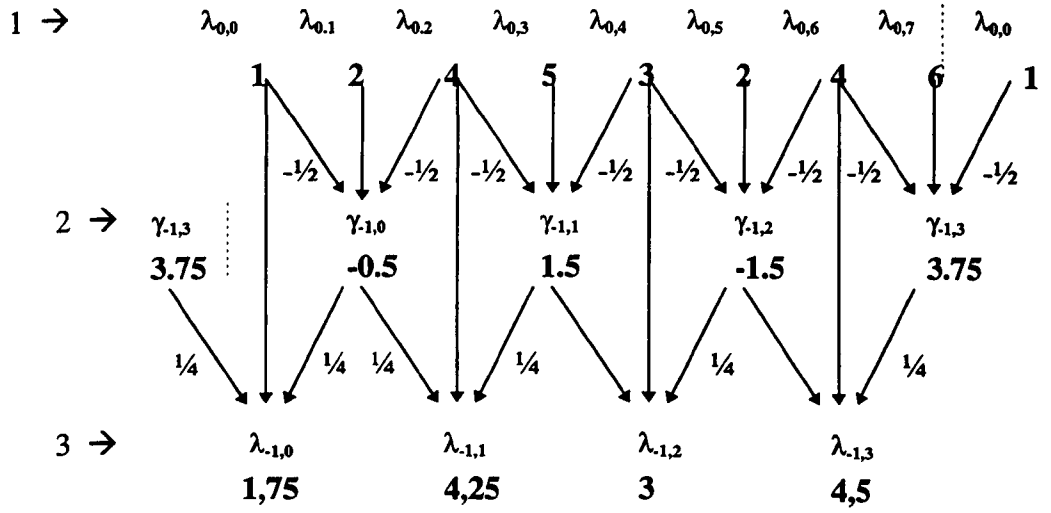


Figure6.15 : Example of the fast lifted wavelet transform with N=1, $\tilde{N}$ =1.

There is now one scale calculated, there are 8 samples, hence a maximum of three scales can be calculated ($8 = 2^3$) on an identical way as before. The amount of wavelet coefficients is divided by two for every subsequent scale, and the last value is the low pass rest $\lambda_{3,1}$ which is the mean of the original signal. If the signal has not a dyadic length (i.e. a power of two), there will be more low pass coefficients.

The calculated low pass coefficient and the wavelet coefficients can be represented in the order as follows, this representation corresponds with the bode plot :

$$\lambda_{-3,0} \mid \gamma_{-3,0} \mid \gamma_{-2,0} \quad \gamma_{-2,1} \mid \gamma_{-1,0} \quad \gamma_{-1,1} \quad \gamma_{-1,2} \quad \gamma_{-1,3}$$

This is the classical organisation of wavelet coefficients. When there is a signal length of 2^n, a wavelet coefficient $\gamma_{j,k}$ is stored in a memory location $2^{n-j} + k$. With a different organisation, in-place calculation is possible with the lifting scheme. The wavelet coefficient $\gamma_{j,k}$ is now stored in location $2^{-j-1} + 2^{-j}k$. This results in the storing of the values, which are in the same column in figure 6.8, on the same location :

$$\lambda_{-3,0} \quad \gamma_{-1,0} \quad \gamma_{-2,0} \quad \gamma_{-1,1} \quad \gamma_{-3,0} \quad \gamma_{-1,2} \quad \gamma_{-2,1} \quad \gamma_{-1,3}$$

6.4.11. Software Realisation

Within the Matlab package m-files were created to apply lifting with spline-filters from which N=1, $\tilde{N}$=1 or N=1, $\tilde{N}$=3. The calculated coefficients were compared with the results of the Wavelab® routines which uses classical methods. The lifting sheme was programmed on a VSP image processing system. Results see Chapter 7.

6.4.12. Conclusions

It may be clear that the lifting scheme is much easier and faster to calculate than the classical fast wavelet transform. In fact, the computed wavelet transform in the example is the same as the $N = \tilde{N} = 1$ biorthogonal wavelet transform of Cohen-Daubechies-Feauveau [54]. The coefficients as found in the example are the same as calculated by convolution of the $\lambda_{0,k}$ coefficients with the filter $\tilde{h} = \{-1/8 \quad 1/4 \quad 3/4 \quad 1/4 \quad -1/8\}$. This convolution needs six operations for one coefficient while lifting with $\tilde{h} = \{1/4 \quad 1$

¼} only needs three operations, a reduction by a factor of two. The reason is that the lifting scheme uses the similarities between the high and low pass filters.
In every step of the traditional transform, the entire signal is low pass and high pass filtered and then subsampled, hence recursion occurs.

Also the in-place calculation can be an important factor when dealing with big data sets.

The inverse transform is very easily put into an algorithm to find by undoing the calculations by changing + into - and vice versa. There are even no other filters necessary in the inverse transform, even this rather is a typical property for biorthogonal wavelets. By comparison, in the classical orthogonal transform it is not so easy to see that the inverse wavelet transform is the inverse of the forward transform.
These improvements make the lifting scheme ideal for time sensitive and memory demanding applications, and in this case for the real-time processing of video signals with the VSP (chapter 7).

CHAPTER 7

Real Time Video Signal Processors

7.1 Introduction

The final goal of numerical algorithms is the implementation on computers and the evaluation of their speed . An important question here is whether or not they can be run in real time. Up till now only software solutions were considered , in this chapter we will evaluate how wavelet algorithms for image processing can be implemented on some very fast processors like the Video Signal Processor (VSP) from Philips and the Multimedia Video Processor (TMS320C80) from Texas Instruments. The architectures will be studied and compared with each other.

On the VSP a noise reduction algorithm with Wiener filtering and based on the lifting scheme will be implemented and evaluated.

On the TMS320C80 a compression based on wavelets and wavelet packets were considered. Variance and maximum entropy were one of the criteria to decide upon the compression rate.

Finally with the possibilities of the VSP in mind, some considerations were made how real time video algorithms could efficiently be implemented in an ASIC (Application Specified Integrated Circuit).

7.2 The Philips Video Signal Processor (VSP)

7.2.1. Introduction

The VSP or Video Signal Processor is a programmable parallel processor designed for real-time processing of video signals. A VSP chip contains a number of processing elements which are able to operate in parallel. It is possible to build a complete video processing system with a set of these multiprocessors. For example, the VSP system used in our application contains a set of 24 VSP1 chips. Vendor Philips currently sells two generations of video processors, the VSP1 and the VSP2.

To program a VSP system, some tools are available which support the design trajectory from video algorithm to code for the VSP's. The algorithms are programmed in a graphical way by schematic entry, there are tools for simulation and the algorithm is mapped on the hardware by interactive and automatic tools.

Programmable hardware has its own advantages when evaluating algorithms. Many different algorithms can be tested and fine-tuned without any hardware modification. Previously developed algorithms can be reused on the same or on similar systems. Algorithms can also be used as building blocks to obtain complex algorithms. However, the greatest advantage is an important reduction in development time. A major disadvantage is the VSP-system's high initial cost.(30000£) .

For future developments, Phideo will come available. This will be a silicon compiler, especially developed for the VSP architecture. Then it will be possible to synthesise the VHDL (Very High Description Language) translation of the algorithm and to place and route the result with Phideo to obtain an ASIC solution.

7.2.2. Architecture of the VSP1 and the VSP2

Real-time video processing is investigated on two different systems : one is developed as a multimedia processor (MVP) by Texas Instruments (TMS320C80) and is available on a PCI board in a Windows NT environment . The other one comes

from Philips (VSP1 and VSP2) and is a parallel construction of VME boards in a rack which communicates with a UNIX workstation.

Texas choose a small number of very powerful parallel processors with 1 master and 4 slaves. (See next paragraph). Philips developed a processor with a much larger amount , but simpler, high-speed parallel processors. The processing is then performed on sample streams and this is the basis for the VSP architecture.

Comparative conclusions will be made at the end of next paragraph.

The sample rates for video signals are in the order of a few to tens of MHz. The handling of these signals requires the integration of high speed communication and sufficient power in a single chip. Each processor executes a small part of the algorithm for all pixels, and therefore each processor has its own, short program. The execution time of a program may not exceed the sampling interval. This can be achieved by compile-time scheduling of the algorithms. The accompanying software tools helped us to do this job.

In the VSP architecture the video signals are processed with a limited resolution of 12 bits. The characteristics of both generations VSP chips are shown in table 7.1

	VSP1	VSP2
technology	1.2 ì CMOS	ì CMOS
chip size	90 mm^2	156 mm^2
transistors	206000	1150000
package	QFP 160, PGA 176	qfp 208
dissipation	1 W	< 5 W
clock	27 MHz.	54 MHz
word size	12 bit	12 bit
ALEs	3	12
MEs	2	4
size ME	512 x 12	2k x 12
memory style	single port	dual port
BEs	-	6

inputs	5	6
outputs (=OEs)	5	6

Table 7.1 : Characteristics of the VSP chips.

The VSP architecture consists of a number of processing elements (PE) connected to a switch matrix. A processing element can be an arithmetic and logic element (ALE), a memory element (ME), a buffer element (BE, only in the VSP2), or an output element (OE). All of these processing elements are pipelined and work active in parallel. The switch matrix realises the programmable communication between all elements in a single cycle.

For each PE the instructions are stored in a local program memory (P). Initially the program memories are loaded with instructions via a serial download. After initialisation, the PEs execute their instructions cyclically. During every clock cycle each PE can take one or more values from the switch matrix. After a few clock cycles the PE will produce a result that is available to all PEs.

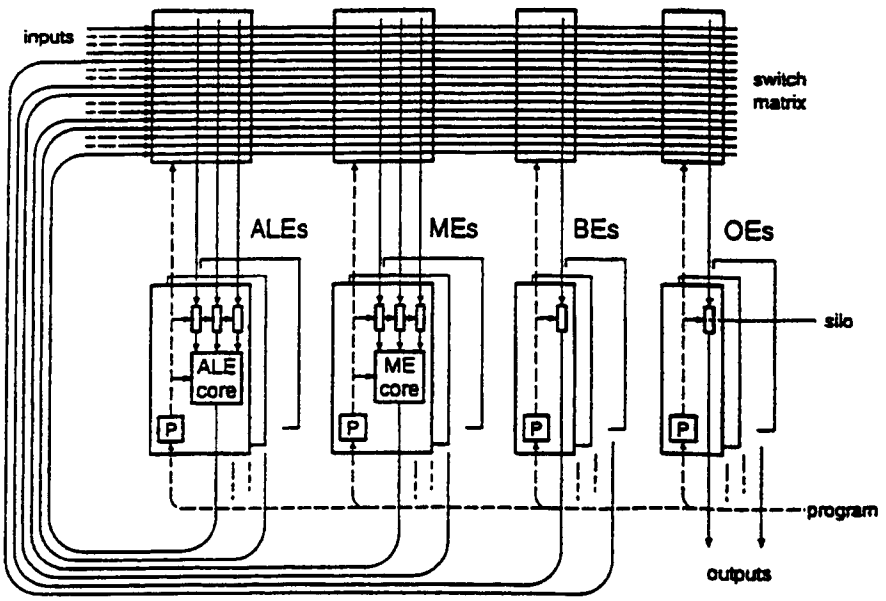


Figure 7.1 : Architecture of VSPs

The silos in a PE are programmable delay elements that are used to store intermediate values. A silo consists of a two-port dynamic RAM and address calculation logic. The storage capacity of a silo is 32 words of 12 bits.

Data from the switch matrix is written cyclically into the RAM. The read address is calculated by subtracting the required delay length from the write address. Due to the cyclic nature and the limited storage capacity of the silo, the maximum lifetime of a value in a silo is 31 clock cycles. The silos are used for the interleaving of a number of data streams in multi-rate processing, and to avoid conflicts on the inputs of PEs.

The arithmetic and logic element (ALE) contains silos, a program memory and an ALE core (figure 7.2). The ALE core contains barrel shifters, multiplexers and an arithmetic and logic unit (ALU). The instruction set of the ALU is given in table 7.2. Multiplication is implemented by booth encoded multiplication steps, avoiding a costly fully parallel multiplier. The instruction set depends on the data. For instance an add1 will either implement the addition of the first two inputs (P+Q), the data inputs or produce a zero (clear), depending on the last bit (r0) of the third input R of the ALU. The instruction set contains a so-called swap bit. Setting this bit makes that the two data inputs P and Q will be swapped after they have been collected from the silo. Consequently, for all instructions a symmetric variant is available.

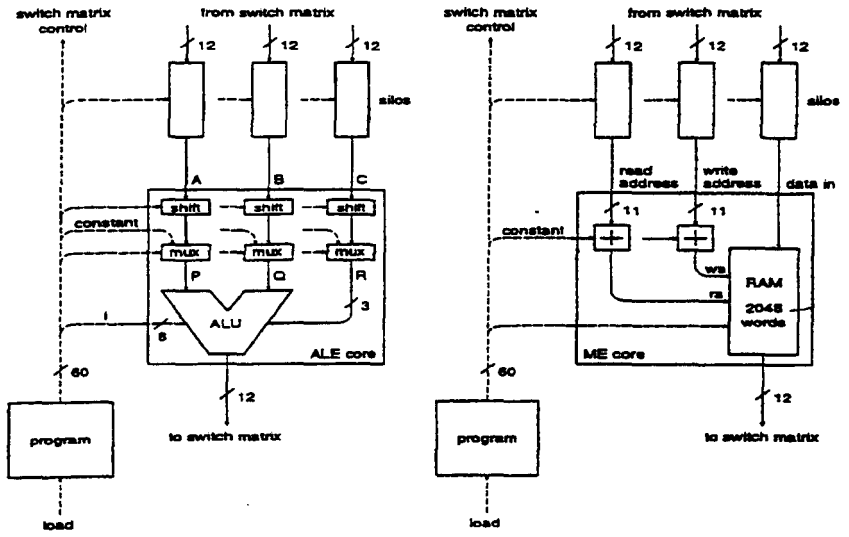


Figure 7.2 : Arithmetic and logic element (left) and memory element (right)

The ALU instruction set.

Name	$i_{4...0}$	$r_2 r_1$	$r_0 = 0$	$r_0 = 1$	Comment
add0	0	x x	P+Q	P+Q+1	
add1	1	x x	P+Q	clear	
add2	2	x x	P+Q	Q	
add3	3	x x	P+1	Q	
add4	4	x x	P+Q	P-Q	
sub0	5	x x	P-Q-1	P-Q	
sub1	6	x x	P-Q	clear	
sub2	7	x x	Q	P-Q	abs(Q)
sub3	8	x x	Q	-P+Q	
sub4	9	x x	Q	P-1	
sub5	10	x x	P-Q	set	set=-1
log0	11	x x	$P \wedge Q$	$\overline{P \wedge Q}$	
log1	12	x x	$P \vee Q$	$\overline{P \vee Q}$	
log2	13	x x	$P \oplus Q$	$\overline{P \oplus Q}$	
log3	14	x x	P	Q	switch
log4	15	x x	clear	set	sign(R)
cmp0	16	x x	$P \geq Q$	true	
cmp1	17	x x	$P > Q$	true	true=-2048
cmp2	18	x x	$P = Q$	$P \neq Q$	false=0
	19	0 0	P	P+Q	2-bit Booth
	19	0 1	P+Q	P+2Q	cell
bm	19	1 0	P-2Q	P-Q	
	19	1 1	P-Q	P	
	20	0 0	clear	Q	3-bit unsigned
	20	0 1	P-2Q	P-Q	multiply (0:7)
um	20	1 0	P	P+Q	
	20	1 1	2P-2Q	2P-Q	
	21	0 0	-P+2Q	Q	3-bit signed
	21	0 1	P	P+Q	multiply (-4:3)
sm	21	1 0	-2P	-2P+Q	
	21	1 1	-2P+2Q	-P+Q	

Table 7.2 : The ALU instruction set of the VSP

The memory element (figure 7.2) contains silos, program memory, data memory and logic for the address calculations. The ME of the VSP1 can implements the storage of parts of video lines, the ME of the VSP2 can store about three video lines. The memory elements can be filled with initial values during program download. This facilitates the usage of memory as look up table (LUT).

The buffer element contains a silo that can be used for the storage of intermediate results. These elements are only available in the VSP2.

The output element provides buffering and the output of the chip. An OE contains only a silo.

To implement complex video algorithms, an arbitrary number of video signal processors can be used in parallel. Field memories are added to the network of processors when required. These memories have to be very fast to meet the speed requirements of the VSP chips running at 27 or 54 MHz.

In the test set-up, three VSP1-flexboards are used (figure 4.3). The VSP1-flexboard is a standard board for the VME bus with 8 VSP1 chips, 4 inputs and 4 outputs. Each square in the hardware-plot below represents a VSP1 with five inputs at the left and five outputs at the right. The connections between inputs, VSP1s and outputs are 12 bits wide.

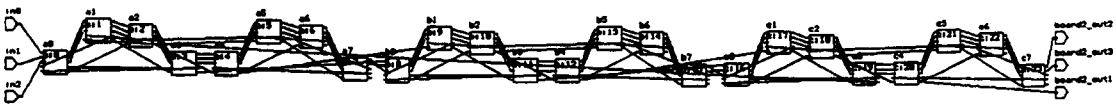


Figure 7.3 : Hardware with inputs, outputs and three interconnected VSP1-boards.

The VSP2 has at least the processing power of eight individual VSP1s. The instruction set is the same in both cases but the VSP2 contains four times the number of ALEs in a VSP1. The VSP2 runs at 54 MHz which is twice the speed of a VSP1. The amount of memory is multiplied by eight in comparison with its predecessor and it also provides for dual ported RAM, enabling a read and a write in one clock cycle. The VSP2 also has 6 buffer elements where the VSP1 has none.

The VSP1 and VSP2 are incompatible at the instruction level but compatibility is provided in the ALU instruction sets and in the programming tools, allowing the designed algorithms to be used on both the VSP1 and VSP2, without the concern for low level compatibility.

7.2.3. Programming of the VSP System

Signal flow graphs are used in order to program the VSPs in a graphical way. These signal flow graphs are inherently parallel : connecting two operations to the same input will cause the execution of the operations to take place at the same time. This is not so easy to program in an existing programming language, such as C. The parallel structure in the signal flow graphs is called fine grain parallelism (for comparison : the TMS320C80 uses coarse grain parallelism). This structure can easily be exploited in the mapping onto a number of parallel processors to achieve the required high level parallelism for real time execution.
Figure 7.6 shows an example of a flow graph with references to the different elements.

The signal flow graphs consist of operations and the arcs (lines) between them are called operands. The operations are elementary operations, selected from the instruction repertoire of the PEs. There are different types of operations : input, output, ALU, read , write, const, offset, shift, delay and pass operations.

- **ALU operations** are arithmetic or logical operations chosen from the ALU instruction set in table 7.2 and they are executed on ALEs. Thus these building blocks perform addition, subtraction, logical AND/OR and so on.
- **Read and write operations** read from or write into data memory of MEs. These memory element operations can be used to implement functions such as line delays and look-up tables.
- **Input and output operations** refer to inputs and outputs of the signal flow graph.
- **Const operations** specify constant values that can be used as input for other operations.

- **Offset operations** specify constant values that are added to the sample values in the data stream passing through. They are useful for the addressing of the memory elements.
- **Shift operations** are logical and arithmetic shifts of the binary sample value. They can be used to perform multiplication or division by powers of two.
- **Delay operations** specify delays of sample streams.
- **Pass operations** can be either buffer elements (only on the VSP2) or output elements. A signal flow graph does not contain the specified pass operations, since these operations play no role at the algorithmic level. However, when the signal flow graph is mapped on a network of VSPs, we usually have to insert a number of these pass operations into the signal flow graph to satisfy the timing and communications constraints. This job is normally performed by automatic tools.

The periodic behaviour of a signal flow graph is specified by the period of the operations. This period indicates how often an operation is executed in relation to the clock frequency. For example, a period of four denotes that an operation needs to be executed once in every four clock cycles. This is for instance at an execution rate of 6.75 if the clock rate is 27 MHz. The operations have different periods in a multi-rate signal flow graph such as in the wavelet decomposition where every lower scale is at half the sampling rate of the preceding scale.

Signal flow graphs can be grouped. This property makes it possible to use them as building blocks to construct more complex hierarchical algorithms.

The programming of a VSP chip requires a number of steps. The automatic mapper supports all steps in the programming trajectory.

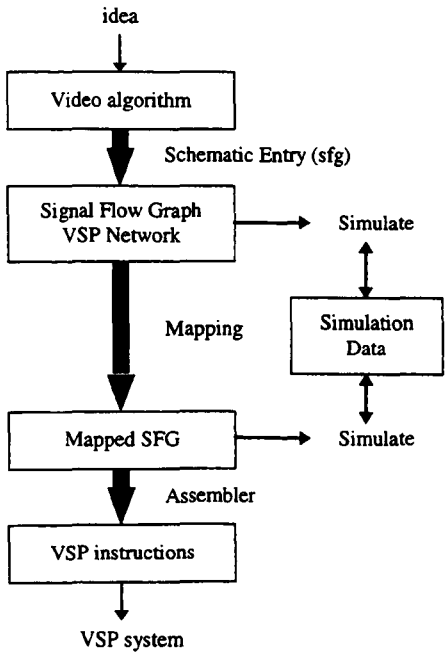


Figure 7.4 : The VSP programming trajectory.

The program part sfg is used for schematic entry of the video algorithm. The algorithm can be simulated with softsim to analyse the functional aspects of the signal flow graph at the level of basic operations. The simulation input and results can be viewed with an oscilloscope-like tool that gives a graphical representation of sample files. The input sample file can be created with a text editor or a UNIX script whereas the output file is generated by the simulator. These files deliver the opportunity to do calculations on them or to compare them with the expected results.

The program part hardware (see Fig. 7.3) is used for schematic entry of the VSP network: The VSP chips, inputs/outputs and the interconnections. However, since VSP-boards are used, a supplied VSP-network can be loaded from a file.

When the simulation results are satisfying, the signal flow graph can be mapped onto the VSP hardware. The program part timemap can be used for the partitioning of the signal flow graph over the VSPs and for the scheduling of the operations in time and over the program memories. Timemap verifies continuously all possible hardware constraints with a direct and interactive feedback, using different colours to indicate violations of constraints. In the timemap window, elements can be changed, added and removed. However, the timemap concept is difficult to understand, time-

consuming and not so interesting since for every small change in the algorithm, the work has be done all over again. A better technique is to use the automatic mapper and to make use of the available options. The mapper likely performs delay management, partitioning and scheduling. Yet mapping is still far from trivial. There exists no set of general solutions and experience is needed to have some feeling with the control parameters.

The last step is performed by the code generation command : the translation of the design into the binary format is loaded in the VSP system.

Figure 7.5 depicts the test set-up with the VSP system. The VSP system itself contains three VSP1-flexboards and an AD/DA converter with RGB in- and outputs. Input sources are a CDI player and a VCR with cable TV or antenna input. The VCR makes it possible to supply a noisy video signal. There is a conversion box for the conversion from scart to RGB (BNC connectors) and vice versa. TV1 is used to view the unprocessed image while TV2 shows the processed image.

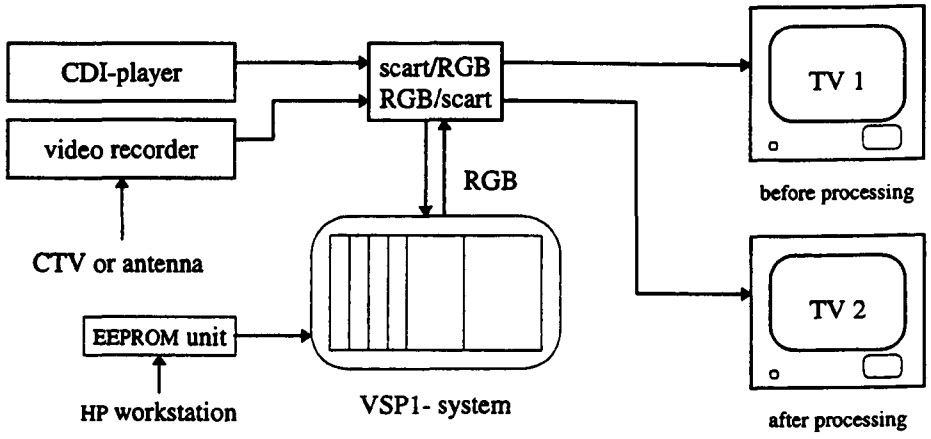


Figure 7.5 : The test set-up.

To make an algorithm working, the appropriate signals (coming from the hardware terminals) has to be supplied to the inputs of the algorithm and its outputs must be connected with the appropriate hardware outputs. Figure 7.3 shows the signal flow graph from the "upper level". The block SFG is a grouped signal flow graph of

an algorithm we want to test. The signal Y_in2 is the luminance signal that determines the brightness of a pixel, thus the information for the black and white image. The luminance signal has an integer range between 0 and 255 (8 bits) representing a scale of . The luminance signal is the most important one since the human eye is most sensitive for this signal, even more than for the colour signal. SYNC_in1 is the video synchronisation signal and does not give any information about the image itself. These two signals has to be combined into one word when they are redirected to the AD/DA board. To obtain this, first of all the luminance signal is clipped into an 8 bits signal by alu0. This ALU performs a bitwise AND operation of the luminance input with the bitwise representation of 255. The synchronisation signal is shifted to the left with 8 bits with the shift operation lsl8. These two signals are added by alu2 and the mixer mix0 performs a demultiplexing, in order to obtain a period of one, needed by the AD/DA board.

There is also the UV_input which is the combined R-B signal. This is thus the encoded and multiplexed colour signal. If necessary this, signal can be changed into an RGB signal. The colour signal is less important for us since the human eye is not so sensitive for colours and the bandwidth of the colour signal is only about half of that of the luminance signal. The colour signal can be set to zero to see the results better.

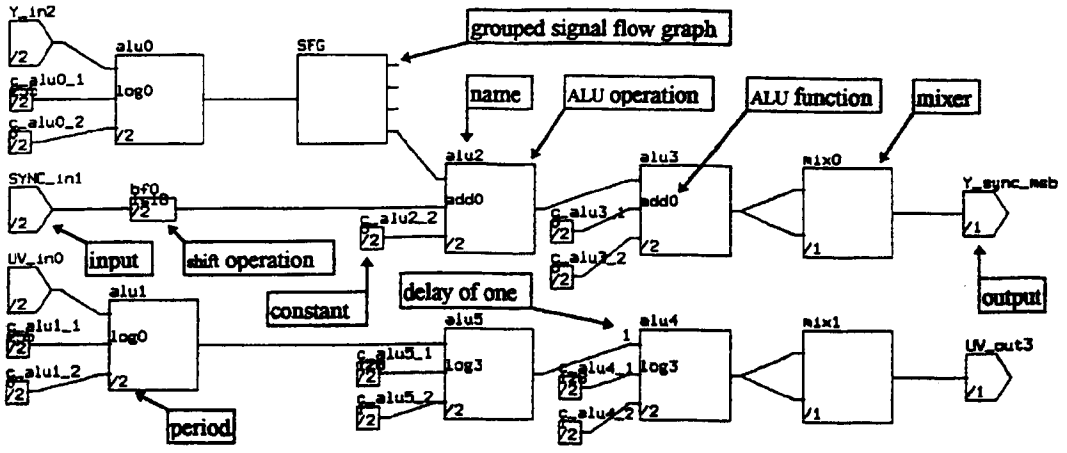


Figure 7.6. Upper level with inputs and outputs. SFG is the processed algorithm.

Remark that, depending on the AD/DA board the input and output signals can be combined in a different signal format. The signals Y, U, V and Sync are mostly multiplexed in the UYVS format, i.e. one U sample by an Y sample, a V sample and an S sample.

7.2.4. Implementation of the Lifting Scheme

Wavelet Decomposition

The signal flow graph is a translation of the algorithm depicted in figure 7.13. There are two highpass outputs; the term outHPx2 was chosen only because the result was twice the one resulting from a Wavelab® analysis. The output outHP may be omitted when mapping .

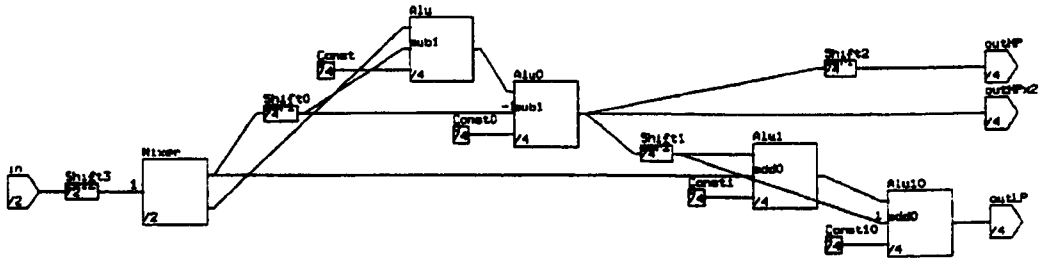


Figure 7 .6. 1 stage in the wavelet decomposition

Notice the "lsl2" at the input, this means a logical shift left with two, thus multiplying with four. This give us an extra precision of 2 bits or, translated into real numbers, steps of 0,25. These shifts has to be omitted at the second level when doing multiresolution analysis. This is necessary because otherwise the signal becomes too large after subsequent decompositions. There are 11 bits available (+1 sign bit), 8 of them are used for the signal, two shift are performed in the first stage, so there is only one bit left over. The extra precision is needed since in the algorithm shifts of two are

used, thus the smallest fraction possible after one stage is ¼. However, two shifts to the left can be to big since the signal grows by maximally ¼ in every step (the worst case input is 0 255 225 255 0). In this case, at black/ white edges overflow is possible, resulting in black pixels in the white regions This seldom occurs in practice. The subsequent steps eliminate the worst case possibility. Nevertheless, it will be explained further that clipping is necessary for some other reasons. It has to be said that when a shift of only one to the left is used, results are as good as with two shifts, despite the fact that there is some information lost. This shows one of the nicest and therefore most used applications of wavelet decomposition : compression or denoising by thresholding the coefficients.

Special attention is needed when using the mixer. Normally, a multiplexer switches over every other sample . VSP-software however, the outputs come available at the same time.

For example, an input sequence

	1	2	3	4	5	6	

results traditionally in

output 1 :	1	1	3	3	5	5	x
output 2 :	x	2	2	4	4	6	6

In the VSP-software however the result is

output 1 :	1	^	3	^	5	^
output 2 :	2	^	4	^	6	^

The sign "^" denote that there is no signal available (the sample frequency is halved). In fact, it is necessary that the two samples are available at the same time, otherwise the samples cannot be subtracted or added to each other. It is easy to

understand that a problem may occur : yet there is a sample at output two before there is a second input sample. In practice, the sequence will be delayed by one sample.

To make it easier to control the algorithm with softsim, it is sometimes preferable to add a delay of one sample before the multiplexer (the "1" at the front side of the mixer). When the algorithm is mapped onto the hardware, the delay can be better left out.

Wavelet Reconstruction

As mentioned before, (chapter 6) while discussing the lifting scheme theory, the inverse transform means simply undoing the steps of the forward transform. One should keep in mind however that, despite we are dealing with biorthogonal filters, the same filters can be used as in the forward transform. Here the negative delay at the end must be omitted if there is no delay at the input of the forward transform.

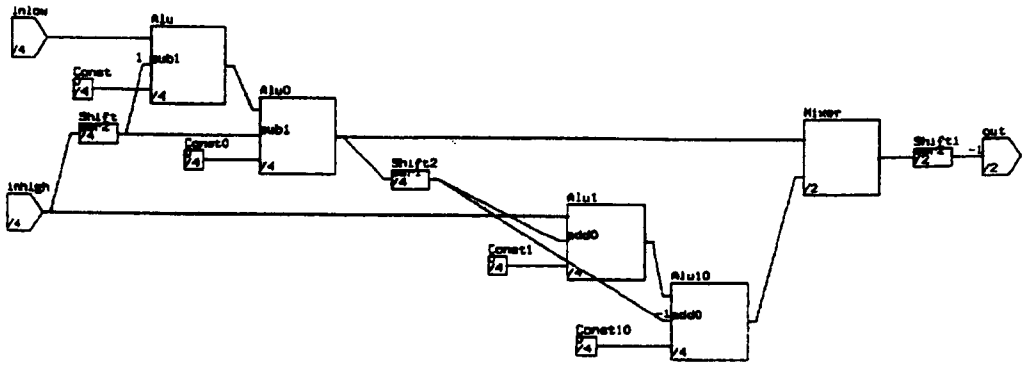


Figure 7.8. 1 stage in the reconstruction

Double Precision Decomposition and Reconstruction

To obtain a perfect reconstruction, even when multiple scales are used, the 12 bit representation of the wavelet coefficients would provide not sufficient precision. For this reason double precision was applied to the wavelet decomposition and reconstruction algorithm. To perform double precision algorithms, a whole set of double precision functions are created, which are also useful in other algorithms. Among these functions we find double precision addition, subtraction, division, and multiplication (by powers of two). They won't be discusssed any further and are made available in a special function library on the VSP system. The forward and inverse wavelet transforms now become :

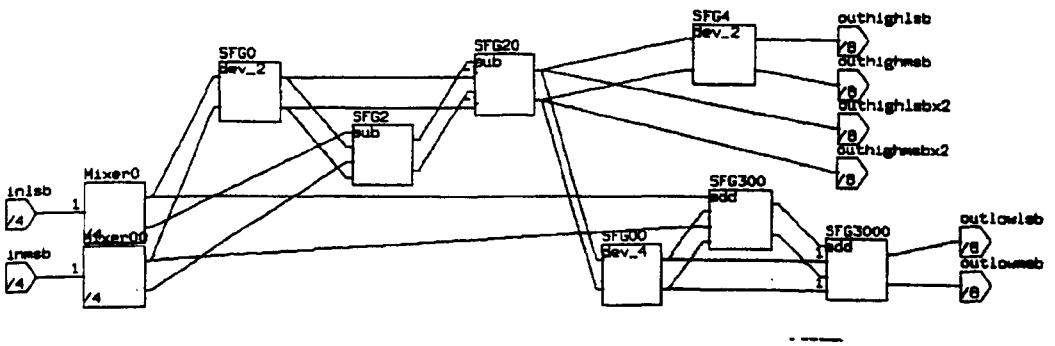


Figure 7.9 : Double precision wavelet decomposition (forward WT).

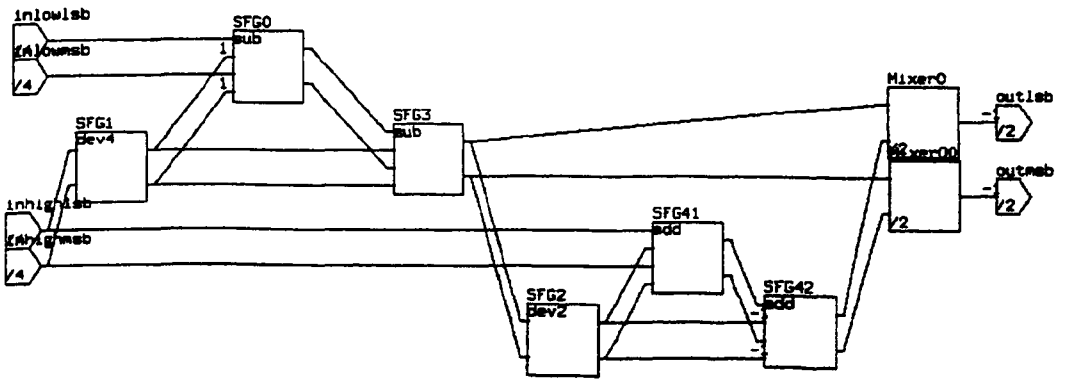


Figure 7.10 : Double precision wavelet reconstruction (inverse WT)

A disadvantage of the double precision solution is that it consumes about three times the amount of ALU's in comparison with the single precision solution. The loss of information in the single precision solution does not give a visible degradation of the image quality when three or less scales are used. Hence, the double precision solution is not further used.

7.2.5 Fixed Attenuation of the Subbands

The following set-up was constructed to investigate the effect of the suppression of the different subbands . No investigation or consideration in the algorithm was, at that moment, made about the S/N (signal to noise ratio) :

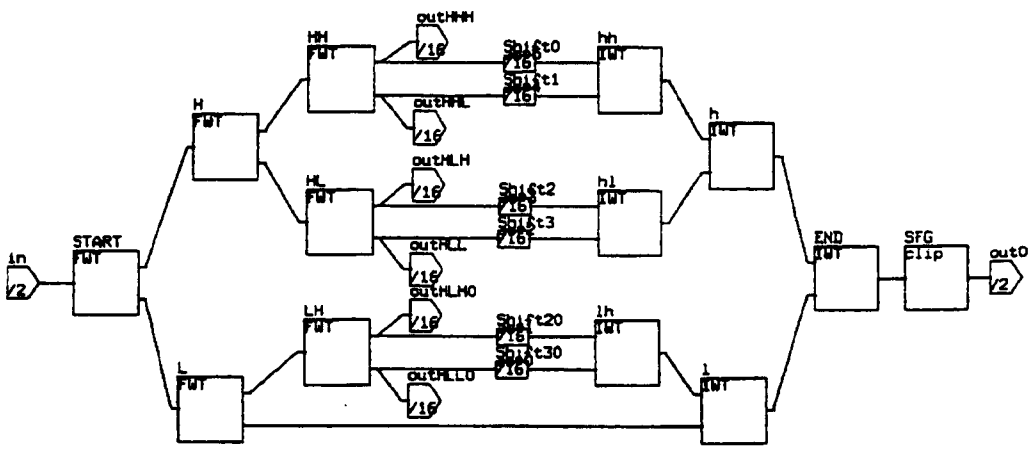


Figure 7.11 : Signal flow graph of wavelet decomposition and reconstruction with fixed attenuation of the subbands.

This accords to the following subband scheme :

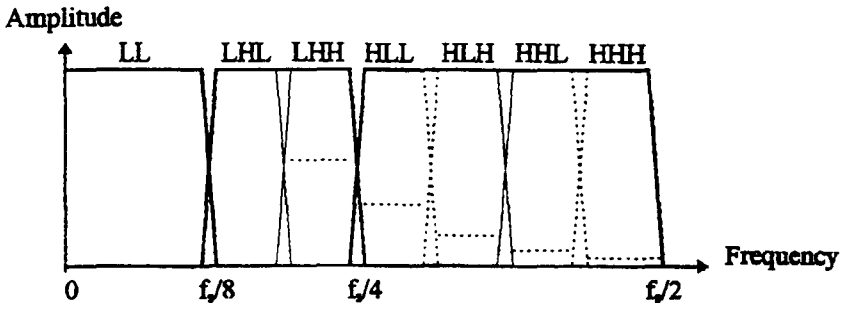


Figure 7.12 : Subband filtering scheme according to fig. 7.11

The thick lines denotes the wavelet decomposition following a dyadic scheme. In the test it turned out that it has no sense to perform the decomposition further than two levels deep. When deeper scales (lower frequencies) are suppressed, too much information is lost, resulting in a extremely low-pass characteristic, with the used filters we obtain jaggy images. The vertical fine and dashed lines denotes further decomposition of the higher subbands. This method is being referred to as wavelet packets decomposition. The filter tree is given by figure 7.13 :

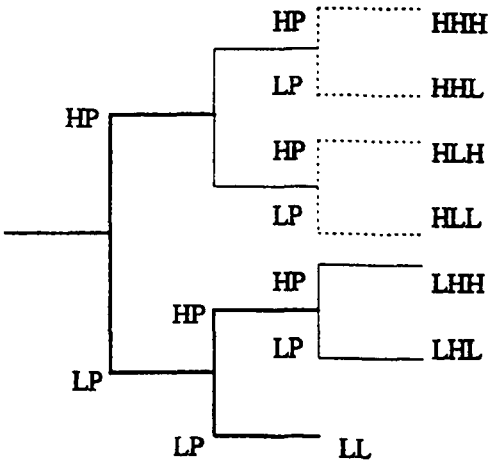


Figure 713: Filter tree according to figure 7.12

The horizontal dashed lines in figure 7.12 denotes the fixed suppression in that particular subband. For the suppression in the higher frequencies, right shifts on the grey level data are used. This results in a dyadic suppression.

Despite the rather rough suppression, the resulting image is as good as the original image in case of a good video signal (CDI-player). Performing the algorithm on noisy video signals (VCR) results in a slight improvement in picture quality (better S/N), depending on the kind of the images. This leads to the conclusion that, as expected, there is more noise than useful information in the higher frequency bands.

So, why not considering a Wiener filtering in wavelet space ? In Chapter 4 some experiments on PC were performed which proved the usefulness of the technique.

Simulation results for VSP can be found in Fig. 7.14.

Note that the suppression on the highest band is performed by 5 arithmetic shifts to the right. However, the highest frequency bands will contain only small numbers due to noise. In fact this results in a complete rejection of the highest frequency bands.

Extensive research should be done on what the "typical" frequency content of a video signal might be considering S/N :

- Is it necessary to subdivide the subbands with wavelet packets? There is a lot of calculating effort in it and maybe the result is not visible (No significant S/N improvement). Scales without important information might maybe omitted totally. In the tests it turned out that the four highest frequency bands could be dropped without significant loss in the picture quality
- VSP1 runs at 27 MHz sampling frequency. This means that 13.5 MHz is the frequency. For video recorders and CDI players this is to high! A subsampling is first necessary so that the Nyquist frequency drops to 6.75 MHz otherwise wavelet decomposition would take place for frequency domains which are not foreseen in the standard of the signal! There are of course other sources with more detail in the picture where the full frequency band is necessary. This is subject to further investigation.

a) the input signal is a block , followed by a peak and a chirp

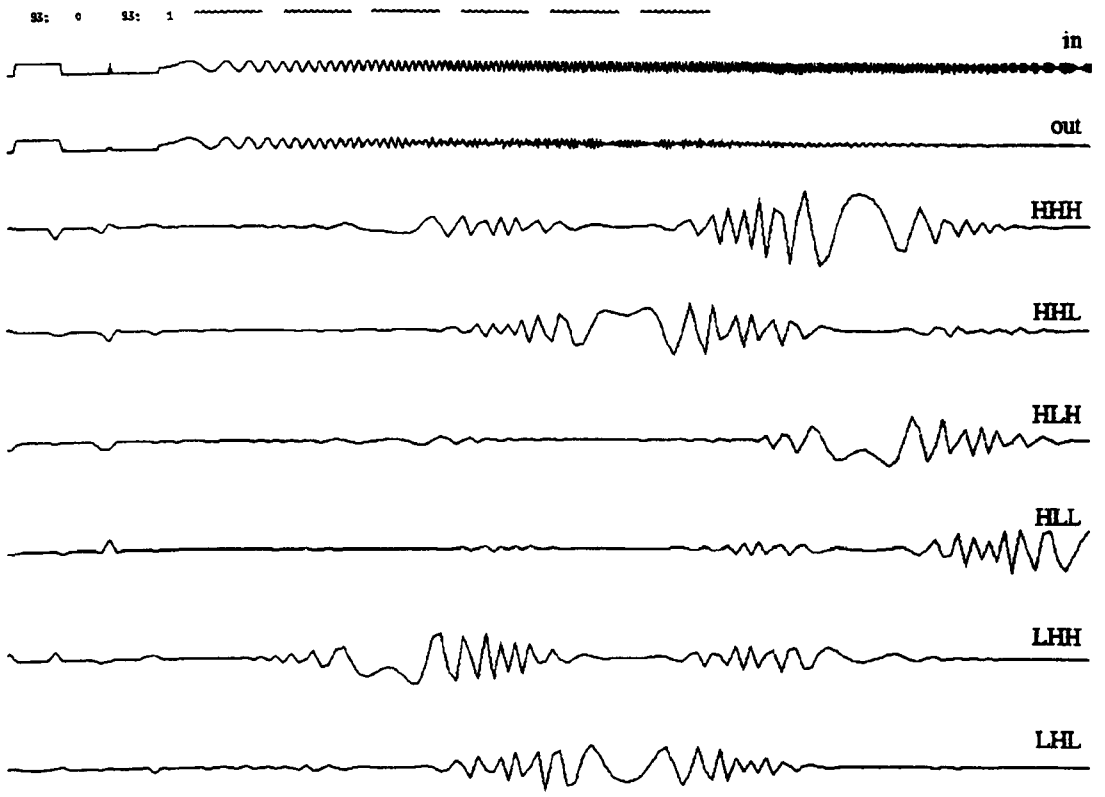
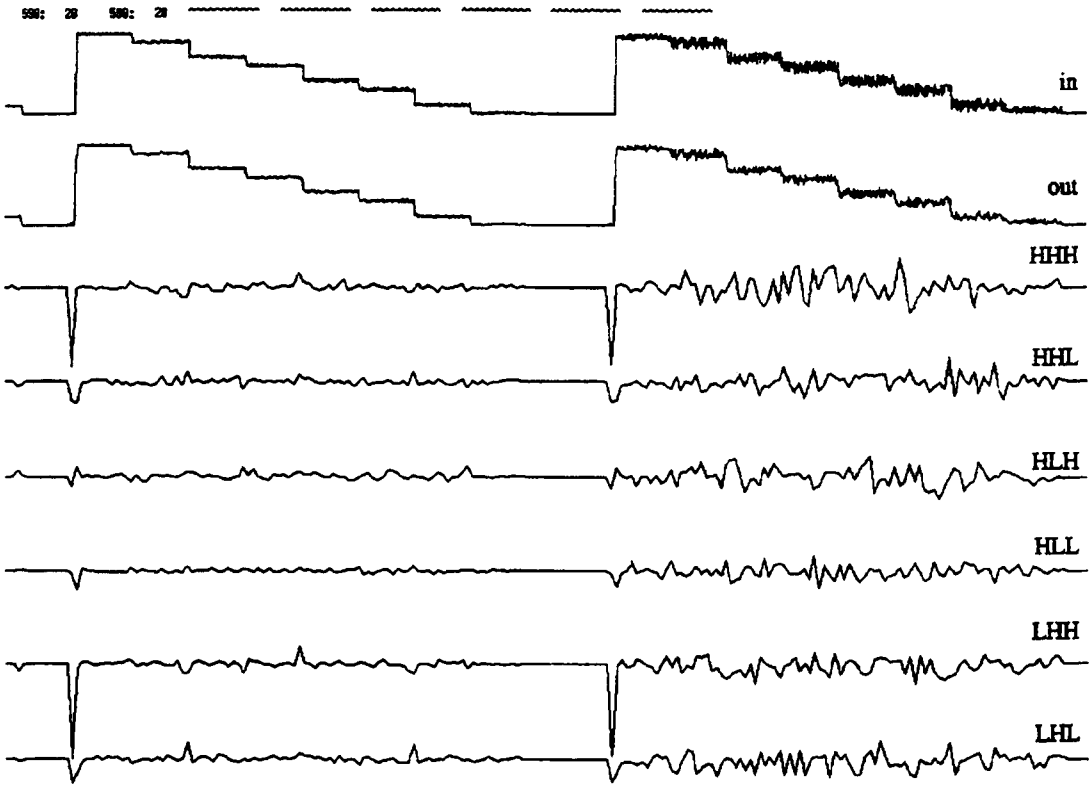


Figure 7.14 a and b (next page) : This is a simulation performed on a VSP station with software developed for VSP1. It is the first stage in a design process before the software is loaded on the VSP boards and run in real time. Figure a does a filtering according to the scheme in fig. 7.12. The high frequencies are filtered out as can be seen in the output signal.

Figure b shows a grey scale pattern with noise at the different levels. The output signal shows a reduction in noise after Wiener filtering in wavelet space using the lifting scheme on a VSP system.

b) The input is an EBU pattern with added noise which amplitude is 10 (on the left) or 40 (on the right)



Notice the clipping at the end of the algorithm. When frequency bands are suppressed or omitted, we get negative and positive peaks after reconstruction. This is because subtraction and addition occur between changed signals. In case of white or black regions with sharp edges in the image, annoying black ripples in white (and vice versa) will be visible. Hence, clipping is necessary. It has however no sense to perform clipping in underlying levels. Remember that the wavelet coefficients in every frequency band have an average of zero, with exception of the lowest frequency band, which wavelet coefficients are the average of the original signal. These frequency bands have an "AC" character with positive and negative values.

The clipping algorithm is depicted in figure 7.15

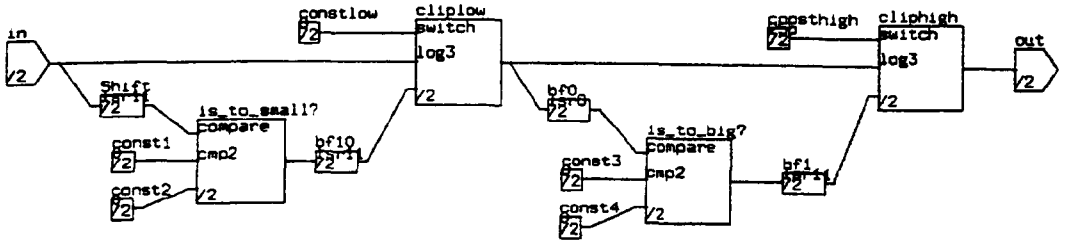


Figure 7.15: Clipping between 0 and 255.

7.2.6. Wiener Filtering

Chapter 4 explored already the problem of Wiener filtering in wavelet space by using Daubechies 4 wavelets. Now we implement the lifting scheme on a VSP1 system.

Determination of the variance of a signal in a subband

The variance of a discrete signal is given by following formula :

$$1/N \sum_{i=0}^{N-1} (x_i - m)^2 \qquad (7.1)$$

With m being the mean of the signal xi and N the number of processed samples. From the wavelet theory we know that all frequency bands have a zero mean, except the lowest band (which is not processed).

Then formula (7.1) becomes $$\sum_{i=0}^{N-1} (x_i)^2 \qquad (7.2)$$

Only the summation of the square of the signal is left. The division by the length of the signal may be omitted if the number of samples used in the determination of the variance would be the same in the different subbands. However, due to subsampling, this is not the case. Decimation halves the number of samples after every step to a new level in the wavelet transform. A right shift (: 2) of the result of form. (7.2) can solve that problem.

Other important questions are :

- How much samples must be processed ? A video line seems to be an appropriate interval for calculations. One should consider that in a real-time process there is a continuous stream of data, so decomposition / reconstruction must take place between successive line windowed samples. Remark that the result of the variance calculation only becomes available at the end of the line. A consequence of this is that the available S/N ratio used to suppress the subbands of an interval in the video signal is in fact the S/N ratio of the preceding interval.
  Concerning this variance calculation two alternatives can be considered :
  - Determine the variance of a short set of samples, and expect the difference in variance with the following set to be small. This will cause problems in the regions on the left of the image, due to line blanking. Extra hardware is needed for this solution.
  - Determine the variance of a whole line. Variance change for subsequent lines are rather small and 'delayed' variances could be used. Nevertheless it requires a lot of memory and there is even extra hardware needed to reject the line blanking parts in the signal. Happily this last requirement is not so badly needed because the 'line blanking error' of the variance is the same at each line and an fixed offset can be subtracted from the variance to get rid of that line blanking error .

- Another problem with the limited set of instructions arises on how to calculate the square in the formula of the variance. There exist a signal flow graph in the library that accompanies the software package which can perform 12 bit multiplication

using six ALUs. Mention that the input value have to be smaller or equal than 45 to avoid overflow (45^2 = 2025d = 7E9h, 46^2 = 2116, 2^{11} -1 = 2047, twelfth bit is sign bit). The signals are however not so small, so a double precision multiplication is needed, requiring more than twenty ALU's, what is very uneconomical.

The idea is now to divide the input by 16 (4 arithmetic shifts to the right) if the signal is bigger than 32 (or 45 as the max), otherwise we let the input stream unchanged. Then the maximum input is 735 since 735/16 = 45,9375 (which is truncated by the shifts to 45). Afterwards, the sample stream is divided by 256 with eight arithmetic shifts to the right (16^2 = 256) if the sample was divided before.

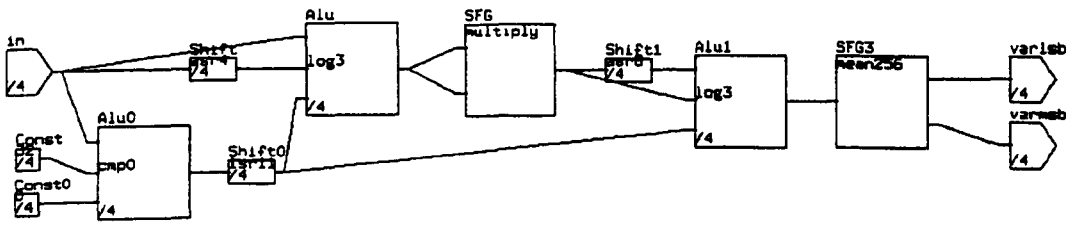


Figure 7.16 : Determination of the variance.

From the formula (7.2) there is still left the summation and the division by the number of processed samples (minus one). This is the same as determining the mean of a signal. At the end of the signal flow graph of figure 7.1 we find the block mean256 that takes the mean of 256 samples in the data stream. The signal flow graph can be found in figure 7.17.

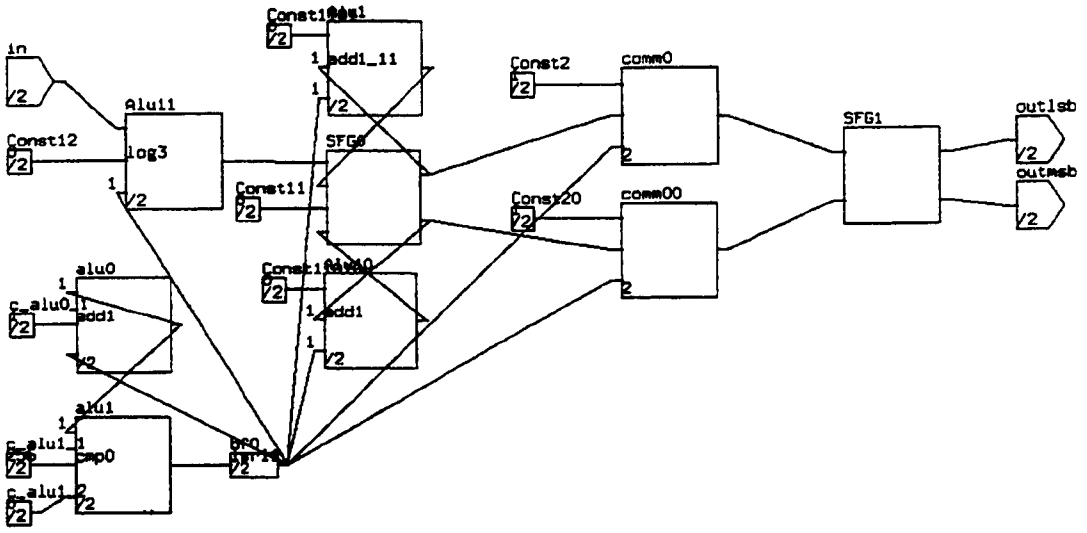


Figure 7.17 : Signal flow graph of the determination of the mean of a data stream.

The amount of processed samples can be chosen by the constant c_alu1, in the corner beneath at the left side of the figure. 256 is chosen as constant , less than a whole line . Considering that the wavelet coefficients of the highest frequency band are at half the sampling frequency of the input signal, there are 512 samples processed. This is yet ¾ of whole line (720 pixels contain information, the other 144 samples are line blanking). Then why not processing a whole line? The answer lies in the fact that we can only divide by a power of two, and 720 and 864 lie between 512 and 1024. We can choose to take the 512 samples in the middle of the screen. Then the rest of the line does not contribute to the determination of the variance.

The summation of the incoming data stream is performed by a double precision addition because otherwise overflow will occur. The output is connected with the second input of the addition with a delay of one. Every 256 samples, the (double precision) result is stored into the memories at the second address (address 1) and the ALU's are reset.

Simple Wiener Based Wavelet Noise Filter

In figure 7.18 a total set-up of the algorithm is depicted. There are two stages in the wavelet decomposition (three frequency bands). The upper frequency band has been totally omitted, hence clipping was necessary. Before determining the variance of the subbands, the mean value of the line is calculated since the mean of the wavelet coefficients in a subband are zero. The variance from the upper subband is subtracted from the variance in the middle subband. Note that the extension _s in sub1_s, which means that the swap bit of the ALU operation is set. The result is compared with the variance of the noise. If the difference is smaller than the variance of the noise, the total signal is omitted, otherwise it will be suppressed in accordance with the S/N ratio.

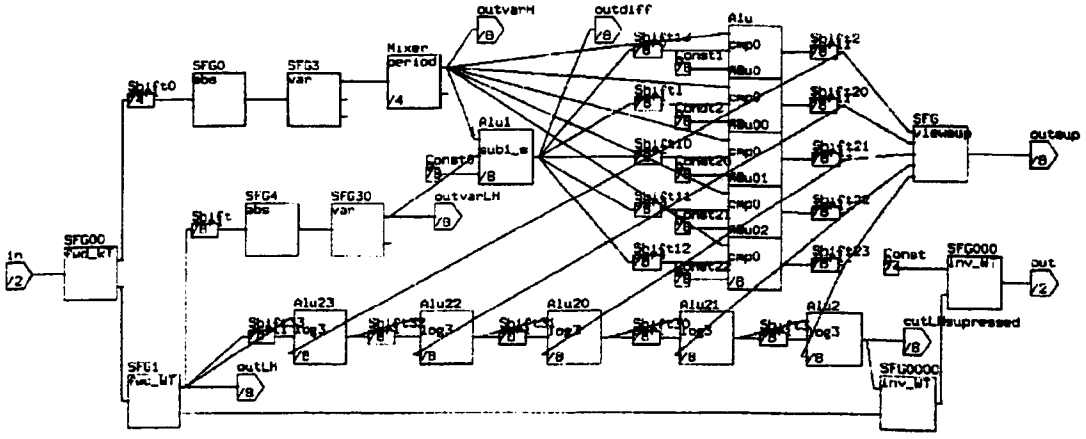


Figure 7.18 : Simple Wiener based wavelet noise filter.

In this set-up, clipping is not included and the logic to calculate the variance somewhere in the middle of a video line is also not depicted.

Despite the fact that this is the most simple set-up, it could not be mapped onto the available VSP system. The factors used in the attenuation part needed to be adapted. More practical experiments are needed to fine tune for better results. We

think however a VSP2 system is necessary to map these algorithms and let them run in real time.

3-Scale Wiener Based Wavelet Noise Filter

To simplify the signal flow graph, the ALUs used to determine the ratio between signal and noise as well as the ALUs used to suppress wavelet coefficients are grouped.

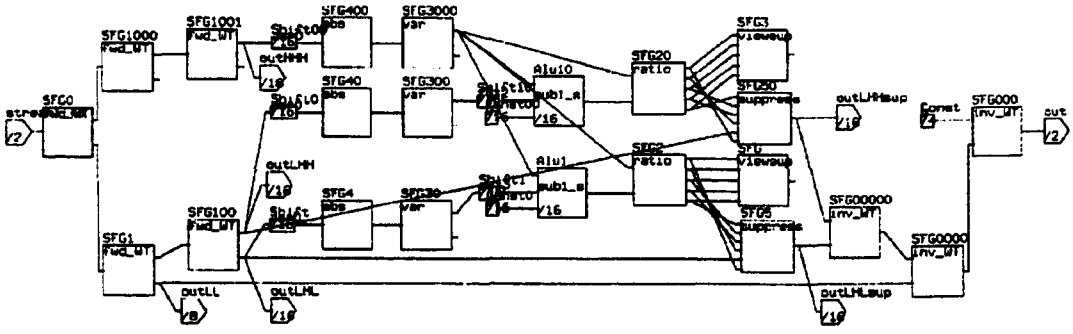


Figure 7.19: 3-scale Wiener based wavelet noise filter

In figure 7.14b you can find simulation results with a noisy EBU pattern as input. The result can be improved by stronger suppressing the lower frequency bands. However, as already mentioned in paragraph 7.5, suppressing these lower frequency bands will result in a blocky effect on the screen.

In the simulation result, there can be seen that there is still much (low frequency) noise in the low frequency band. Wiener filtering is only a primitive way of working in wavelet space, maybe it should be combined with other techniques like wavelet transform maxima and multi-scale edges described by Mallat [65].

7.2.7 Other Methods

The following techniques were partially examined or realised.

Thresholding of Wavelet Coefficients

Thresholding wavelet coefficients is a method often used to denoise images. Hard thresholding is easily implemented. The difficulty is to find a 'good' threshold because it depends on the content of the image. Wiener filtering can be used in combination with a thresholding dependant on the S/N . The results were not satisfactory (see Par. 7.4)

Median Filter

The median of a sequence is the middle value of the sorted sequence. A median filter is useful to eliminate impulsive noise and has the property to maintain the sharp edges. Good results can be obtained when applied to images deteriorated with speckle noise. But as the video signal is more contaminated by Gaussian noise, the results are not so good.

Local Maxima in Wavelet Coefficients

Mallat [65] proposes to represent a signal by its **wavelet local maxima** at all levels. He says :" Look at the wavelet where something interesting is happening." This technique gives good results for denoising and encoding.

7.2.8. Comparison and Conclusions

The algorithms for the wavelet decomposition and reconstruction while using the lifting scheme proved to work very well.

The fixed attenuation of the subbands gave us a first glance of the result of the suppression of certain frequency bands. Note that the quality of the filtering is rather subjective and depends strongly on the content of the image.

The variance can efficiently be determined with a minimum amount of ALUs.

The complete Wiener based wavelet filtering could only be simulated with the available VSP system. Contacts were made with NATLAB Eindhoven (Philips Headquarters) to continue the experiments on a more powerful system (VME rack with VSP2 boards in stead of VSP1).

Some research is also done by one of my MSc students on how to implement VSP algorithms into an ASIC. Phideo is the name of the silicon compiler to do so. I won't further report on these research fields because this is in fact no more directly related to non-stationary signal processing but more on how to implement it in silicon.

Finally, we must conform that it is difficult to find objective criteria to measure image improvements .Signal-to-noise ratio improvements can be made impressive but , as already mentioned in chapter 4 they very often confirm nor deny of what you really see!

7.3 The Texas Instrument TMS320C80

7.3.1. Introduction

The TMS320C80 is a single-chip parallel multimedia video processor (MVP) developed by Texas Instruments. It is the flagship of the TMS320 family of digital signal processors (DSPs) and the first in the series that has multi-processing possibilities. It can be used for applications such as image-processing, virtual-reality graphics, audio/video digital compression, image recognition, . . .

The TMS320C80 integrates onto a single integrated circuit; five fully programmable processors, a direct memory access controller with a DRAM, SRAM and VRAM external memory interface, 50KBytes of SRAM and a video timing controller. Five of the four processors are identical advanced digital signal processors (ADSPs) supporting fixed point operations. The fifth processor, the master processor, is a 32-bit RISC CPU that includes an IEEE-754-compatible floating point unit. All processors are both programmable in assembler and C. The transfer controller is an intelligent DMA controller that manages all memory traffic. Packet transfer can be performed between on- and off-chip memory. This includes instruction- and data-cache services as well as byte-aligned array transfer. The TMS320C80 is capable of performing two billion RISC-like operations per second. During each second of processing it can move 2.4 Gbytes of data and 1.8 Gbytes of instructions within the chip, plus 400Mbytes of data to off-chip memory.

The TMS320C80 is sometimes referred to as the MVP (Multimedia Video Processor) or abbreviated to C80.

7.3.2. Some Key Features

1. More than two billion operations per second

2. Four parallel processors (32-bit ADSPs with fixed point unit)

One master processor (32-bit RISC with IEEE-754 floating point unit)

3. 50K of on-chip RAM

4. Memory capabilities:
   - Five instruction fetches per cycle (1.8 Gbytes/s)
   - Ten parallel data accesses per cycle (2.4 Gbytes/s)
   - The 64-bit transfer controller is capable of up to 400Mbytes/s on- and off-chip memory transfers.

6. Dynamic bus sizing for 64, 32, 16 or 8 bit-access to VRAM, DRAM and SRAM.

7. A video controller with dual frame timers for simultaneous image capture and display.

8. Four external interrupts, edge- and level-triggered.

9. Full-scan design (plus a boundary scan) testing via an IEEE-1149.1 Test Access port.

10. Operating voltage: 3.3 V.

11. TI EPIC 0.5/0.6-mm CMOS technology.

12. Approximately four million transistors.

13. 305-pin ceramic PGA package.

7.3.3. System Architecture

An overview of the C80 system architecture can be seen in figure 7.20.

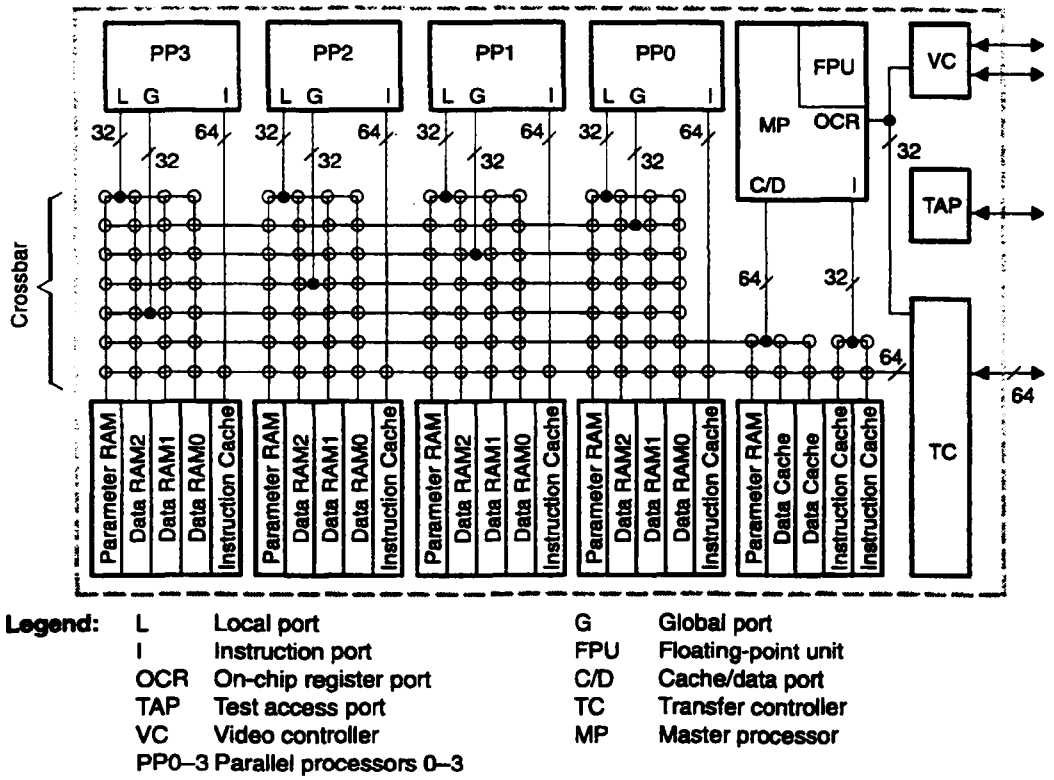


Figure 7.20 Block diagram of the TMS320C80

7.3.3.1. The Master Processor (MP)

The master processor is a 32-bit RISC processor with an integrated IEEE-754 floating point unit. The MP is structurally designed for efficient execution of C code. The floating point instructions are pipelined; therefore, you can start a single-precision multiply or any floating-point add instruction on each clock cycle. Floating-point unit operations use the same register file as the integer and logic unit. A scoreboard ensures that correct register-access sequences are maintained. MP instructions and data are fetched from on-chip caches, each of which is 4Kbytes in size. The control for these caches is an integral part of the MP design. The MP is able to access the on-

chip memories by using the crossbar network.

There are three active low edge-triggered interrupts, EINT1 - EINT3 (EINT1 having the highest priority), that allow external devices to interrupt the master processor. The EINT3 also serves as an unhalted signal that causes the MP to fetch its reset vector. The C80 also contains an active low level triggered interrupt LINT4.

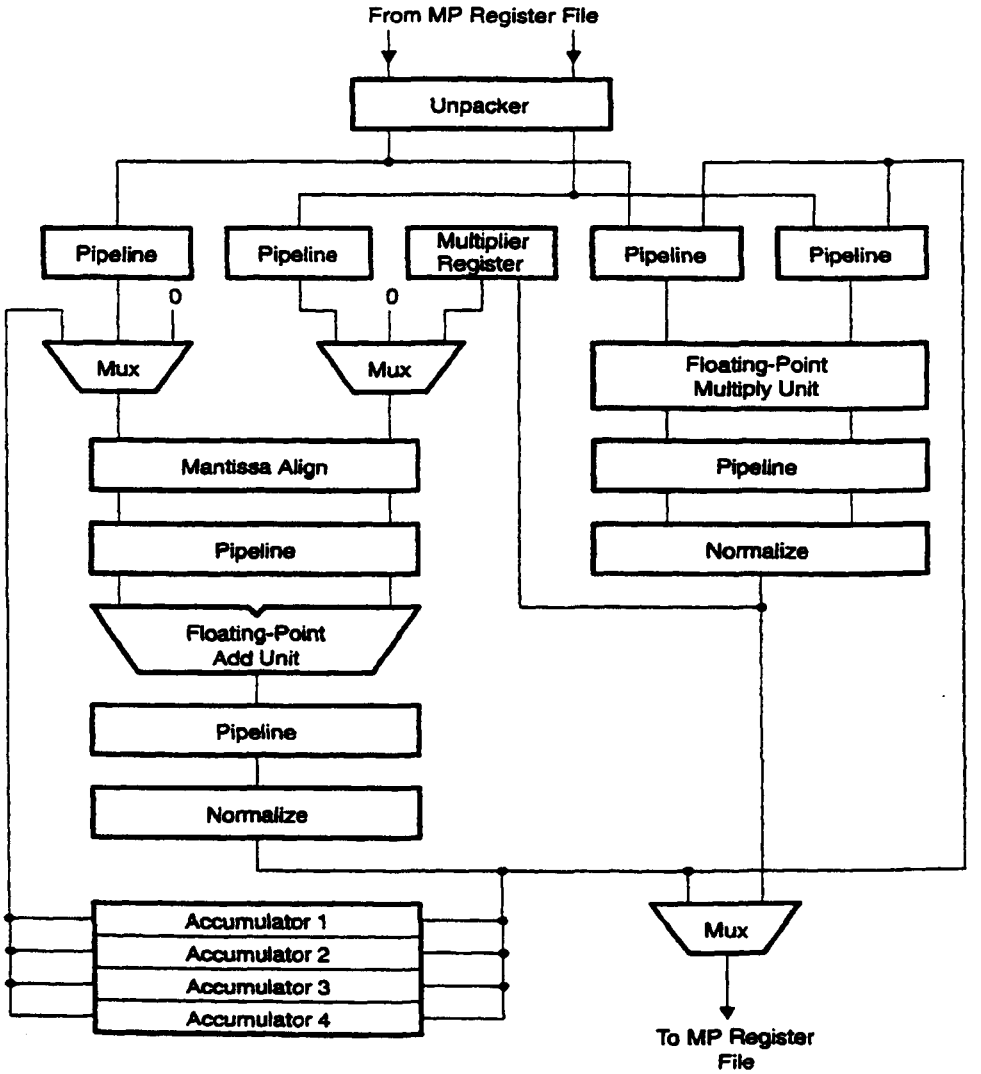


Figure 7.21 General floating-point flow

The Floating Point Unit

This unit is capable of performing IEEE-754 floating point operations in 32-bit single-precision (type float in C) or 64-bit double precision (type double in C). Conversion between different formats is also supported. It also provides vector floating point operations to improve program efficiency. These vector operations are pipelined, so you can start both a floating point multiply and a floating point add on every cycle.

Hardware support for the floating point unit consists of a full double precision floating point add unit and a 32-bit single precision multiply unit:

1. The add unit handles additions, subtractions, compares and conversions through the following stages:
    - compare exponents and shift smaller number right to align binary point
    - add/subtract two numbers (two's complement notation)
    - normalise an output value
    - round an output value
2. The multiply unit handles multiplies, divides and square roots through the following stages:
    - double or single precision floating point multiplication
    - normalisation and round of the output value

Floating point operands are read from the register file when an instruction is dispatched. Execution typically takes more than one cycle. The result is stored back into the register file when the instruction completes. A register scoreboard maintains correct access sequences.

MP Instructions

The MP has three basic instruction formats: short immediate, three-register and long immediate. Following types of instructions are provided:

1. Arithmetic, logical and compare instructions
   - integer add and subtract
   - logical instructions
   - compare instructions

2. Floating point and vector instructions
   - floating point arithmetic and conversion instructions
   - vector floating point arithmetic, multiply, add/subtract and conversion instructions
   - double precision floating point accumulations in vector instructions

3. Program-control and context-switching instructions
   - branch unconditionally
   - branch conditionally, compare to zero
   - branch conditionally, branch on bit
   - call functions, subroutines and return

4. Control register instructions
5. Leftmost and rightmost one instructions
6. Load and store instructions
7. Shift instructions

7.3.3.2. The Four Parallel Processors (PPs)

The Advanced DSPs

The four advanced digital signal processors (ADSPs) operate under the command

of the master processor. Each ADSP can perform digital signal processing, along with bit-field and multiple-pixel manipulations. These processors can perform in excess of ten RISC-like operations in each cycle.

In figure 7.22 the block diagram of an ADSP is shown. The diagram shows three large blocks: the data unit, the address unit and the program control unit. Each of these blocks will be elaborated in the paragraph below.

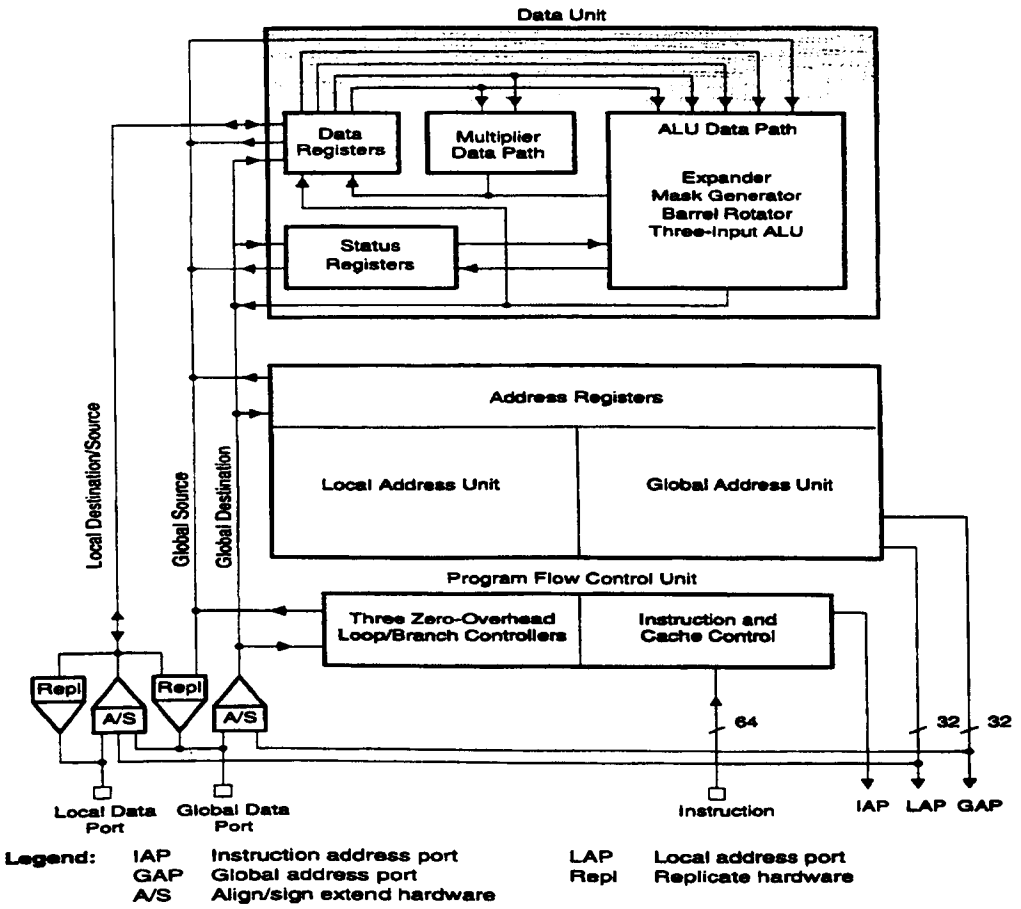


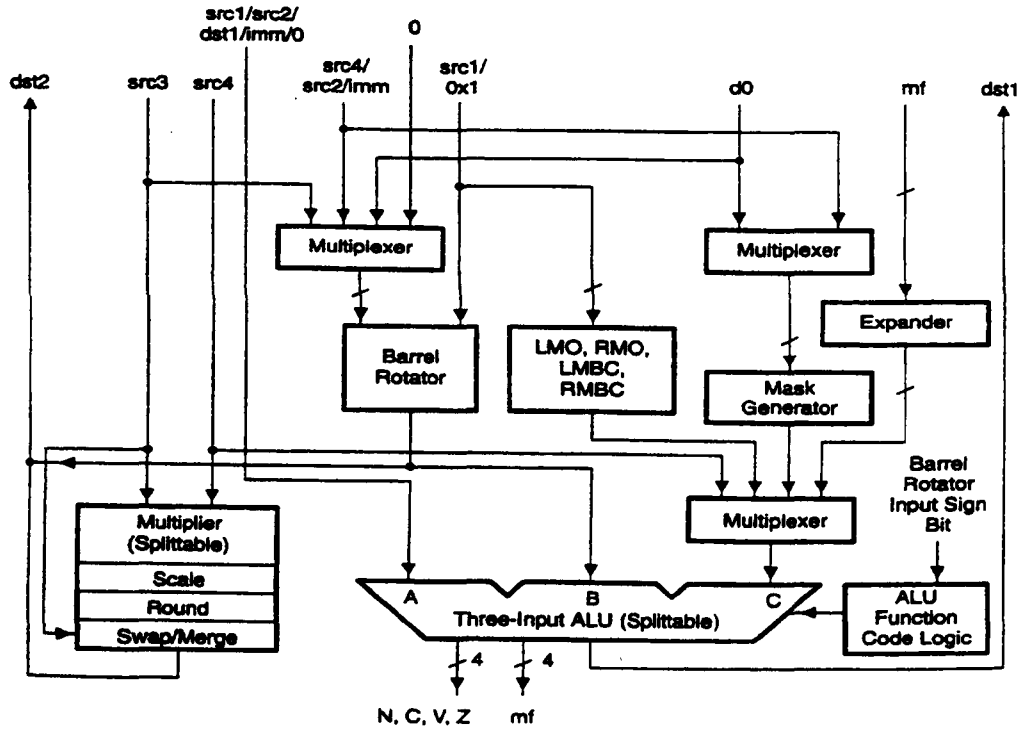
Figure 7.22 The ADSP block diagram

The Data Unit

The data unit is composed of three major elements: the data unit register, the multiplier data path and the ALU data path. Figure 7.23 shows the data unit's block diagram (not including the data unit registers).

Figure 7.23 Multiplier and ALU data paths

In addition to the ten registers, the unit has two functional independent data paths; the multiplier data path and the ALU data path. The multiplier data path includes a 16-



by-16-bit multiplier, a half word swapper, and scaling and rounding hardware. The ALU data path includes a 32-bit three-input ALU, a barrel rotator, a mask generator, a multiple flags expander, and logic to detect the bit number of the leftmost-one, rightmost-one, left-most-bit-change, or right-most-bit-change in a register.

The ten registers on the unit are eight data registers (d0-d7), a status register (sr), and a multiple flags register (mf).

The Address Unit

Each ADSP has two address units; a global address unit and a local address unit. An instruction can specify up to two independent memory accesses, one by each unit. The address units can perform register-to-memory stores and memory-to-register loads. They can also perform general-purpose data computations referred to as address unit arithmetic. This capability, along with conditional loads of the program counter register, speeds up functions that are computation- or jump-bound rather than memory-access limited.

The Program Flow Control Unit

The program flow control unit controls the ADSP instruction pipeline, fetches and decodes instructions, performs any necessary handshaking with the transfer controller, and handles interrupt response and prioritisation.

The major hardware elements of the program flow control unit are:

1. **The Instruction Controller**

   It takes 64-bit instructions that the cache controller fetches and generates the control signals that drive the ADSP. Instructions are processed by a pipeline that consists of three stages: an instruction fetch, address unit computations, and data unit execution.

2. **The Program Counter Registers**

   Consists of:

   - The program counter (pc); points to the next instruction to be fetched
   - The address-stage instruction pointer (ipa); track the program counter
   - The execute-stage instruction pointer (ipe); track the program counter
   - The return-from-subroutine instruction pointer (iprs); saves the return-address for a call operation

3. **The Cache Controller**

   This controller compares the instruction address as given by the pc and determines

if the instruction is in the ADSP's cache. If the instruction is already in cache, the controller translates the address into the location in the ADSP's cache RAM and issues the translated address over the instruction address port (IAP). If the cache controller determines that the instruction is not in cache, it issues a request to the transfer controller and stalls instruction pipeline until the cache has been loaded with the new instruction.

4. **The Three Zero-Overhead Loop Controllers**
   Support up to three simultaneous hardware controlled loops. Since each ADSP instruction performs so much in parallel, key loops often require very few instructions, as such this allows for even nested loops to have zero loop-control overhead.

7.3.3.3. The Transfer Controller (TC)

The transfer controller (TC) is a combined DMA (direct memory access) machine and memory interface that intelligently queues, prioritises, and services the data requests and cache misses of the five programmable processors in the C80. Through the transfer controller all of the processors can access the system external to the chip. In addition, data-cache or instruction-cache misses are automatically handled by the transfer controller. In figure 7.24 you can see an overview of the transfer controller:

1. The independent source controller and destination controller process the source and destination addressing.

2. The PT FIFO (packet transfer first-in first-out) supports DRAM page and burst modes and buffers between byte-misaligned accesses to access memory more efficiently.

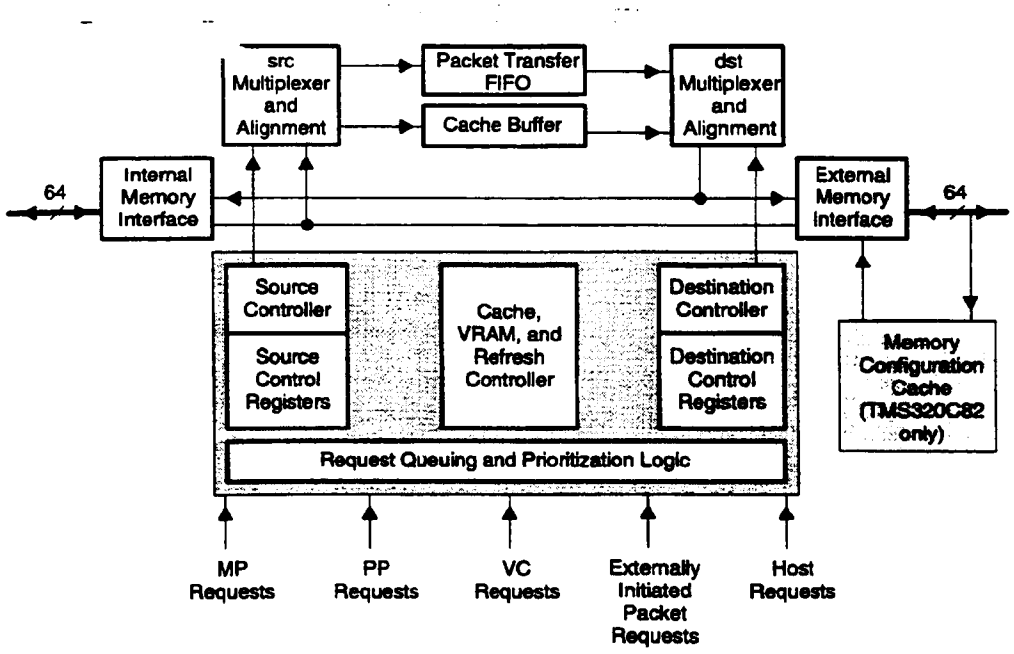


Figure 7.24 Transfer controller block diagram

3. A separate cache controller can preempt program-controlled packet transfers to service cache misses. This controller uses the cache buffer to buffer incoming data.

4. The request queuing and prioritisation logic prioritises the active requests and starts transfers. The transfer controller automatically suspends and later resumes lower-priority requests when a higher-priority request occurs.

The transfer controller facilitates access to the off-chip memory interface. This interface supports dynamic RAM, video RAM, static RAM, and ROM.

Data transfers are specifically requested by the ADSPs or the MP in the form of linked list packet transfers, which are handled by the transfer controller. These requests allow multidimensional blocks of information to be transferred between a source and a destination, either of which can be on-chip or off-chip. Transformations from X/Y arrays to linear arrays can be done autonomously, which improves the efficiency of processing by the ADSPs or the MP. The transfer controller supports a number of

packet transfers (moving blocks of data form source memory to destination memory) which includes:

- **Dimensioned transfers** can be used to copy blocks of data and to manage algorithms that process data by rows, columns, or MxN blocks.
- **Fixed-patch guided transfers** are those in which the sequence of dimension addresses is guided from an on-chip memory-table, rather than calculated solely from values within the packet transfer parameters. Fixed-patch guided transfers (using an on-chip guide table) come in three types: delta-guided, offset-guided, and offset-guided look-up table (LUT). Variable-patch guided transfers (not using packet transfer parameters) can be either delta-guided or offset-guided.

The TC also supports a host-interface mechanism that allows a host processor (in this case a Pentium 133) to gain access to the C80 system, through direct access to all the memory control signals.

Dynamic bus sizing in configurable on a page-by-page memory basis enabling the selection of 64-, 32-, 16-, and 8-bit memory transfers.

7.3.3.4. The Video Controller (VC)

The video controller is the interface between the C80 and the image capture and display systems. The video controller provides simultaneous control over two independent frame systems and two frame memories. The frame systems provide screen resolution and data capture. The two frame memories are memory region coordinated with the systems as either frame grabber of frame buffer image storage. A block diagram of the VC is shown below.

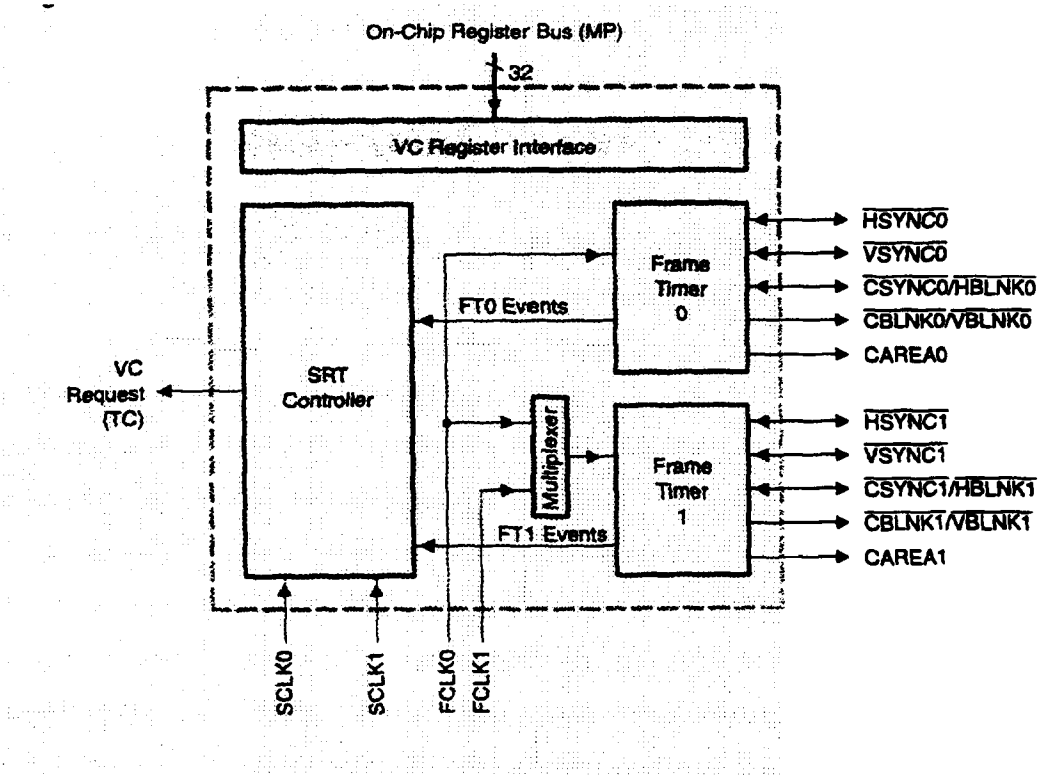


Figure 7.25 Video controller block diagram

The video controller has following features:

- Two identical frame timers provide video timing control
- A serial register transfer controller (SRT) generates SRT cycle requests to the transfer controller to transfer data into and out of VRAM.
- A register interface is accessed by instructions from the master processor
- The multiplexer allows synchronisation of the two frame timers to FCLK0.

7.3.4. The TMS320C8x Software Development Board

The Software Development Board (SDB) is a PC/AT plug-in card that allows you to develop software for the TMS320C8x. The board is connected to the PC using the PCI bus (Rev. 2.0 or 2.1). It integrates video capture, processing, video display along with audio capture and playback.

7.3.4.1. Architecture

Figure 7.26 illustrates the SDB system architecture, the physical interface between the host CPU and the TMS320C80 is the PCI FIFO and PCI controller, which is represented as one block called the host interface. The SDB memory controller provides the interface between the 32-bit PCI FIFO, the 64-bit TMS320C80 data bus and the 16-bit I/O bus. It also controls the peripheral data transfers (PDT) between the PCI FIFO and either DRAM or VRAM memory and it controls reads and writes to devices connected to the I/O bus. The memory consists of 8 MBytes of DRAM for code and data storage and 2 MBytes of VRAM for video display to an RGB monitor. The video capture function is implemented on a daughtercard that is responsible for capturing, decoding and scaling the video signal. An audio CODEC supports capture and playback of CD quality audio.

To develop software in a practical environment, processing tools are supplied. The tools run under Window NT and communicate with the SDB through a specially made device driver.

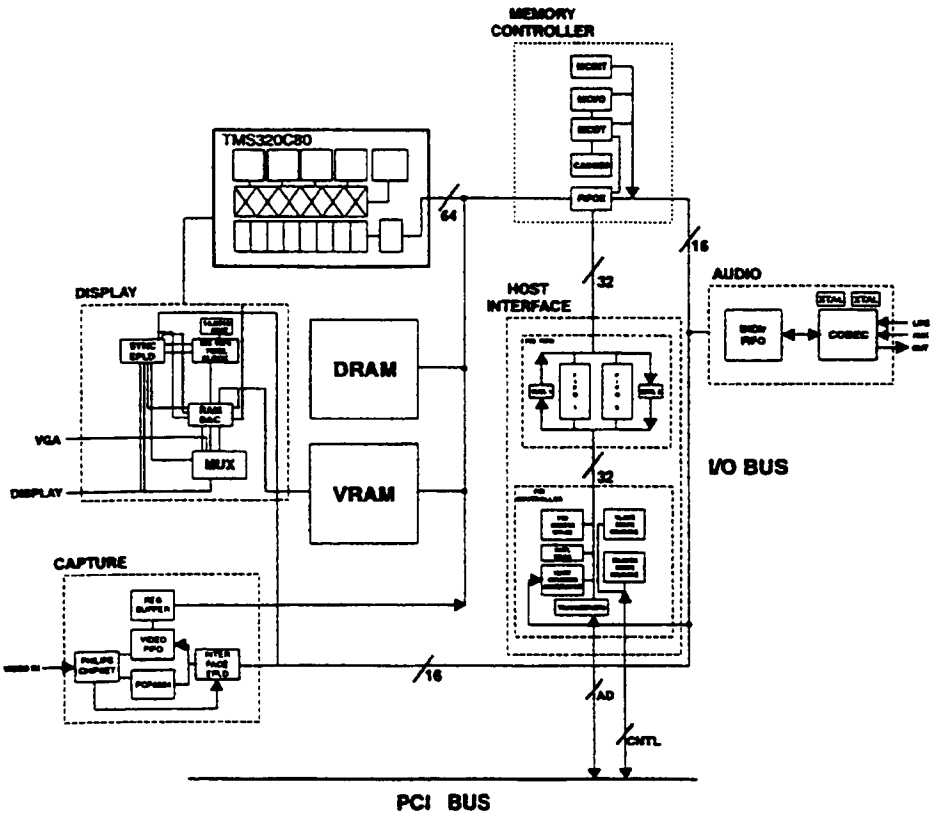


Figure 7.26 Block diagram of the software development board

The Functional Blocks

The SDB has six major functional blocks (as seen on figure 7.26).

The functional blocks include:

- MHz TMS320C80
- Host interface block
- Memory controller and DRAM/VRAM
- Video capture block
- Video display block
- Audio capture and playback block

TMS320C80 Processor

The TMS320C80 chip was developed by Texas Instruments as one of the most efficient and highest performance digital signal processors available on the market today. For a description of this chip see chapter 1.

Host Interface

The host interface is the interface between the host computer, a PC/AT, and the SDB. The access is processed through a PCI bus (Specifications Revs. 2.0 & 2.1). The maximum peak rate is 132 Mbytes per second. All communications between the SDB and the PCI bus is routed through a 32-bit wide by 64-word deep FIFO (first in, first out register). The FIFO also contains mailboxes that are used for single data transfers in both directions.

Five communication methods are implemented:

- Host access to the internal PCI interface registers
- Host access to the SDB register space without using the TMS320C80
- Host access to the SDB through block transfers (BLTs)
- Bus mastering from the SDB to the PCI
- Host access to the entire SDB address space

Memory Controller

The memory controller allows the connection between the TMS320C80 and the external memory and the other system resources in an efficient and easy-to-use manner. It uses pipelining to ensure that the C80 spends little time waiting. The memory controller supports the programmers need to emulate interrupts in hardware and to detect the current condition of the interrupt pins, as well as the status of exceptions.

Thus, the two main responsibilities of the memory controller are:

- Interfacing the C80 and the other external system resources
- Taking care of interrupts and interrupt conditions

Video Capture

The video capture block, illustrated in figure 7.27 uses the following components:

1. The Philips chip set video digitizer/decoder:
   - TDA8708A analogue-digital input interface
   - TDA8709A analogue-digital input interface
   - SAA7196 digital decoder/scaler/clock generator
2. A 512x64-bit video FIFO
3. A 64-bit register buffer
4. An interface EPLD
5. The PCF8584 parallel-serial I²C-bus controller

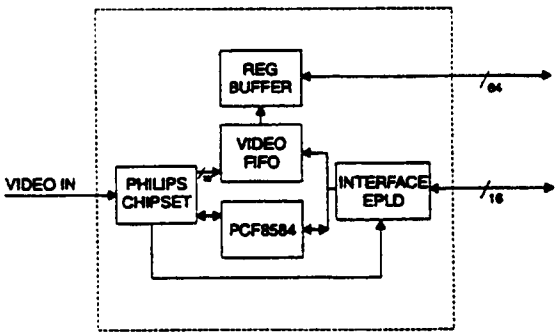


Figure 7.27 The video capture block

The process for capturing, storing, and processing video input always begins when a NTSC or PAL video signal is sent from an external source such as a VCR or a camera in either composite or S-VHS format. The incoming video signal is digitised by the TDA8708A and the TDA8709A 8-bit A/D input interfaces. The video signal is digitised into an 8-bit Y/C (luminance/chrominance) format or into an 8-bit CVBS format, depending upon the video input type. The digitised video signal is then decoded and scaled by the SAA7196. The supported scaled output formats are:

- 16-bit 4:2:2 YUV
- 24-bit 8:8:8 RGB

The scaled video data are buffered in a 16 x 32-bit output FIFO register onboard the

SAA7196.

The SAA7196 FIFO outputs are directly connected to the 512 x 64-bit video FIFO and the 64-bit register buffer. The video FIFO's input is controlled by the interface EPLD. If one line of video data is captured, one line of data is output to the SDB data bus, under the control of the TMS320C80 and the memory controller.

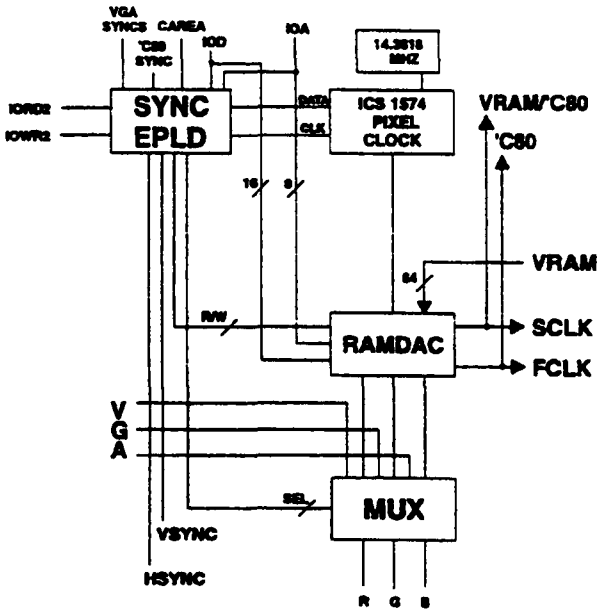


Fig. 7.28. The video display block

Video Display

The video display block, illustrated in figure 7.28, uses the following components:

- A TVP3020 video interface palette
- A QS3257 analogue multiplexer
- Mbytes of VRAM
- An Altera 44-QFP sync EPLD EPM7032
- A ICS1574 user programmable oscillator pixel clock generator

The display circuitry supports resolutions up to 1600 x 1200 x 8 bits per pixel and a 60Hz refresh rate. The timing for a given resolution and refresh rate is established using the programmable pixel clock generator (85 MHz) and the RAMDAC (170 MHz). The pixel clock drives the RAMDAC, which then generates the serial clock (SCLK) for the VRAM ,the C80 and the frame clock of the C80. The frame buffer stores RGB pixel data that is converted to analogue signals by the TVP3020 video interface palette. The display memory provides a single 2-MByte frame buffer of 256 x 1K x 64 bits using four 4-Mbit VRAMs each with an internal organisation of 512 x 512 x 16 bits. The analogue MUX and sync EPLD allow the display to be synchronised to an external VGA signal.

The supported resolutions are:

- 640 x 480 x 8 bpp 75 Hz
- 640 x 480 x 24 bpp 75 Hz
- 1024 x 768 x 8 bpp 75 Hz
- 1024 x 768 x 16 bpp 75 Hz
- 1280 x 1024 x 8 bpp 75 Hz
- 1600 x 1200 x 8 bpp 60 Hz

Audio capture and playback

The audio capture and playback portion of the SDB consists of a bidirectional audio FIFO, a DMA interface and an AD1848 audio CODEC. The FIFO size is 1024x16 bit and the capture and playback implements 16-bit stereo.

7.3.4.2. Code Generation Tools

Generating Code

To generate code to run on the SDB, a memory map file must first be defined for the linker. This file contains the addresses of all the available resources. The SDB has five memory areas: DRAM, VRAM, capture FIFO, I/O, and PCI FIFO.

Then, the assembler or c-code for the program is written. Using the C80 at full use, a

minimum of five code-files has to exist: a master processor source code file and four parallel processor source code files (one for each PP). The shell programs 'ppcl' (for the PPs) & 'mpcl' (for the MP) compile, assemble and optionally link in one step. The format is:

ppcl (-options) (filenames) (-z (link options) (object files))
mpcl (-options) (filenames) (-z (link options) (object files))

The compiler includes a parser, an optimiser and a code generator. If the COFF object is generated, everything is linked to form an executable object file.

File name extensions determine the file type :

- .asm or .s* : assembly language source file
- .c or no extension : C source file
- .o* : object file
- .out : executable file

Loading and executing code

The output files from the C80 code generation tools are in common object file format (COFF). The C80 COFF files always contain the following four sections:

- MP code section (text)
- PP code section (ptext)
- Predefined constants (const)
- Initialised data section (data)
- Uninitialised data section (bss)

To load a COFF file into SDB memory, it must be parsed to identify the individual sections and the addresses in which they are to be loaded. The SDB provides two methods of loading and executing code: using the emulator and its associated debuggers or using the SDBshell, which is a Windows NT application.

The Shell and Debugger

The SDBshell is started through the command 'sdbshell'. Programs are executed by typing the command 'exec' followed by the name of the executable file (a .out file). If e.g. the memory map is incorrect a 'COFF' file error is generated. For more information about such an error the debugger has to be started. The debugger is shown on the screen as five separate windows, one for each processor. Programs can be trace trough c code and assembler, to find the bugs.

7.4. Wavelet Compression

7.4.1. Definition

Compression is a technique for reducing the size of a block of data that represents a signal/image, while keeping as much valuable information about that signal/image as possible It in fact decorrelates the data of all redundant information so that only the essence of the signal/image stays. The reasons for this can be :

- storing more information in a certain memory.
- transmitting more information in the same amount of time.

Major question of course is how to decorrelate valuable from worthless data

There are two kinds of compression techniques:

1. **Lossless** Compression

   The reconstruction of the information results in the original information

2. **Lossy** Compression

   When the information is reconstructed the resulting information in not the same.

It is obvious that lossless compression can be used everywhere. Lossy compression can only be used in certain domains. E.g., performing a lossy compression on a computer program source or execution code, would result in a useless file. Lossy compressed images or sound, on the contrary, could still result in

subjective good reconstructions.

An example of lossless compression is coding the most frequent number with less bits and coding less frequent numbers with more bits. The result will very likely be a compressed series. Nevertheless, a header containing the information about which number is coded with which bit series must also be included. The inclusion of an information header is always necessary and reduces the compression ratio.

Examples of lossy compression techniques are JPEG, MPEG , fractal compression, . . . The subject of this paragraph, wavelet compression, is also an example of lossy compression.

The compression ratio is defined as the ratio between the size of the original information and the size of the compressed information. If N is the number of samples in the original series and M is the number of samples in the compressed series, the compression ratio (CR) is:

$$CR = \frac{SIZE_{original}}{SIZE_{compressed}} = \frac{N}{M} \tag{7.3}$$

The compression ratio is sometimes expressed as a percentage.

7.4.2. Cost Functions

If an image is transformed by the wavelet transform, it results is a group of subbands. These subbands contain information of different parts of the spectrum. Some subbands will contain more information than others. It is thus logical to conclude that some subbands are more important than others. So, neglecting the less 'important' subbands will result in a lossy compression. Still, an information header will have to be included containing the information about which subbands are, whether or not, included in the compressed image. With wavelet compression, a distinction will have to be made on using wavelet packets or normal wavelets.

A big problem in this reasoning is the word 'important'. That is, which subband contains information valuable enough to keep and which contains neglectable one. A very subjective question of course! Nevertheless, computers have to use objective criteria. Some of the criteria are discussed in following paragraphs.

7.4.3. Thresholding

A primitive reasoning could be that a subband contains important information about an image if the coefficients in that subband are large. Two methods could be considered :

- Take the sum off all samples in all the subband and discard those with the smallest sums.
- Look for the large numbers in all subband and discard all those smaller then a certain level.

Both methods were examined , and both showed rather poor results. The second method seemed a bit better than the first, because edges in high frequent subbands result in spikes and not necessarily in a large sum.

7.4.4. Variance

When a signal contains a lot of change , one could say that it contains a lot of important information. The amount of change can be expressed through the variance

$$\sigma_x^2 = \sum_{n=0}^{N-1} (x_n - \bar{x})^2 \qquad (7.4)$$

of a signal.

This criteria is used in noise reduction. The use of this cost function resulted in much better results than thresholding.

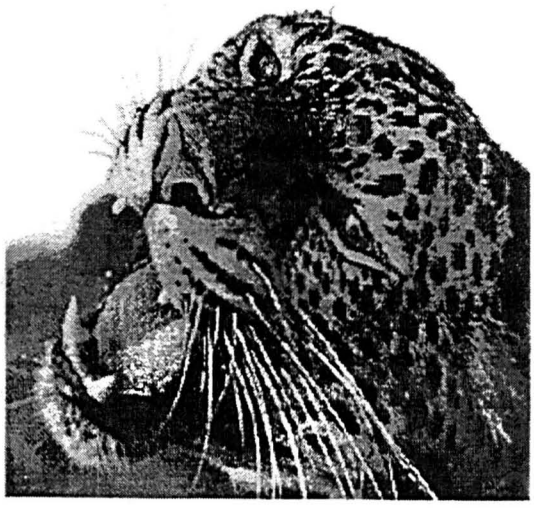
7.4.5. Entropy

Entropy stands for the amount of chaos in a signal. In some literature the entropy is defined by using the probability densities. We prefer not to use the probability densities and take the simpler expression :

$$H = -\Sigma^N x^2 . \log(x^2) \qquad (7.5)$$

This cost function gave approximately the same results as the variance cost function.

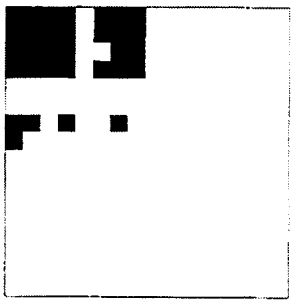
7.4.6. Results



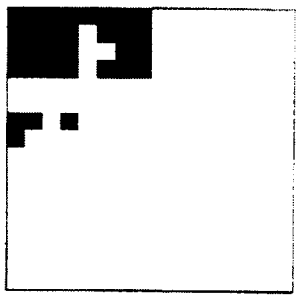
(a)

(c)

(e)



(b)



(d)

Fig. 7.29. The leopard's head is compressed and reconstructed using wavelets. The cost function are : for (c) the variance and for (e) the entropy. Figures (b) and (d) indicates in black which parts of the wavelet analysis LL part (see Fig.4.21) are significant enough (based on variance in b and on entropy in d) to be synthesised. The non-significant coefficients are made zero.

In figure 7.29, you can see the difference obtained with the variance and the entropy cost function. Remark that only one subband is different. The compression method used was: 2D wavelet packets with Daub12 on 16x16 pixels and a fixed compression ratio of 8.

(a) Original image 256x256

(b) Wavelet Packets used in the reconstruction (c) (Variance)

(d) Wavelet Packets uses in the reconstruction (e) (Entropy)

7.4.7. The Ripple-Effect

As can be seen in figures 7.29 (c) and (e), there is an annoying ripple in the background. This is the result of too much neglects in certain subband. The reason for the ripple-effect is illustrated clearly in figure 7.30. If a heavy edge is sent through a wavelet highpass and lowpass filter, overshoots and undershoots can be noticed. When the highpass spikes are neglected the resulting image has an annoying ripple. This manifests itself in pictures where the black:white transient is rather abrupt. Black, having the value zero, is changes to a negative value just before and after the transitions. But, when a negative value is converted to a byte (the images are 8 bpp) this number is truncated to a very high positive value resulting in a white line. This can be seen in figure 7.31. The multiple lines occur when the wavelet transform is

performed until a certain level. This white lines problem was solved by changing negative numbers to zero, in the last level.

Nevertheless, these ripples can be unpleasant to the eye, but certainly much less annoying than the squares occuring when an image is compressed using JPEG.

There are some methods to reduce these ripples. Mostly they include iterative procedures with a lot of iteration (10 to 30). So, not very interesting in real-time applications.

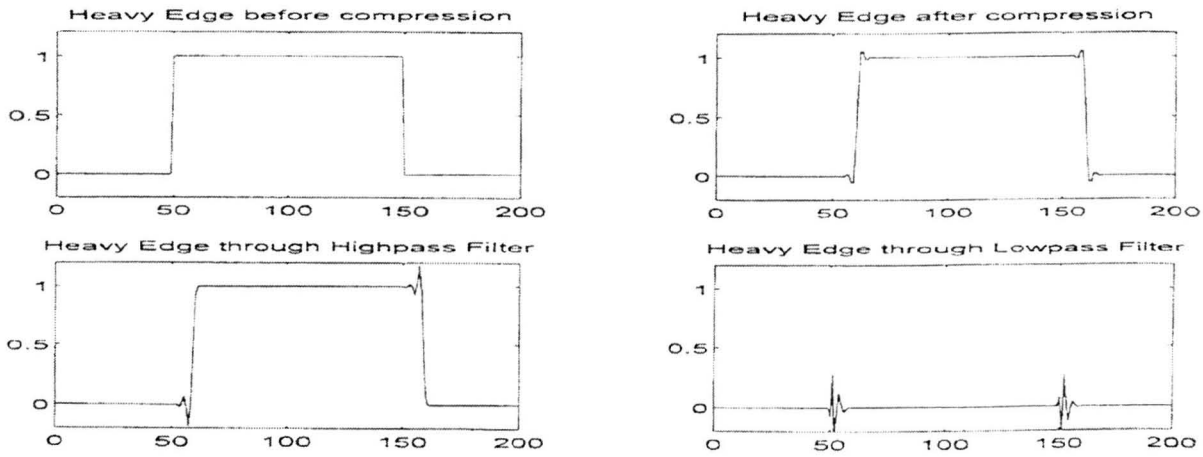


Fig. 7.31. Origin of ripple effect

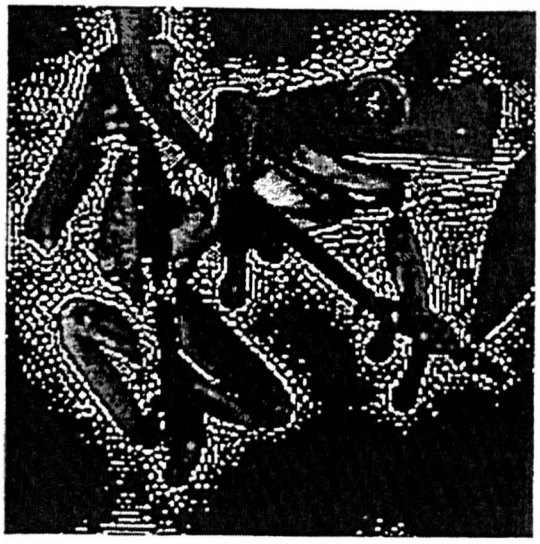


Fig. 7.32. White ripples caused by discontinuous black to grey transition

7.4.8. Parameters in Wavelet Compression

Quality of wavelet compression is dependant on many different parameters, all contributing to a better or worse compression of the image. Some of these parameters are:

- wavelets or wavelet packets.
- number of stages.
- kind of filter (Daubechies, Biorthogonal, . . .).
- order of the filters (Haar, Daub6,12,..).
- cost function (Thresholding, Variance, Entropy, . . .).
- level of transform.

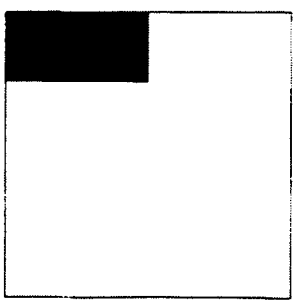
A few of these will now be investigated.

7.4.9. Wavelets vs. Wavelet Packets

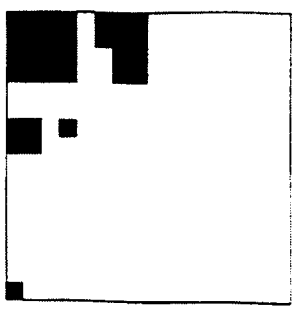
In figure 7.33 wavelets are compared to wavelet packets. With wavelets there are less calculation involved. wavelet packets however have the advantage of having more subbands to select coefficients from.

A disadvantage of wavelet packets is that the ripples are more pronounced. On the other hand they lead to the construction of 'best' bases based on cost functions [67]

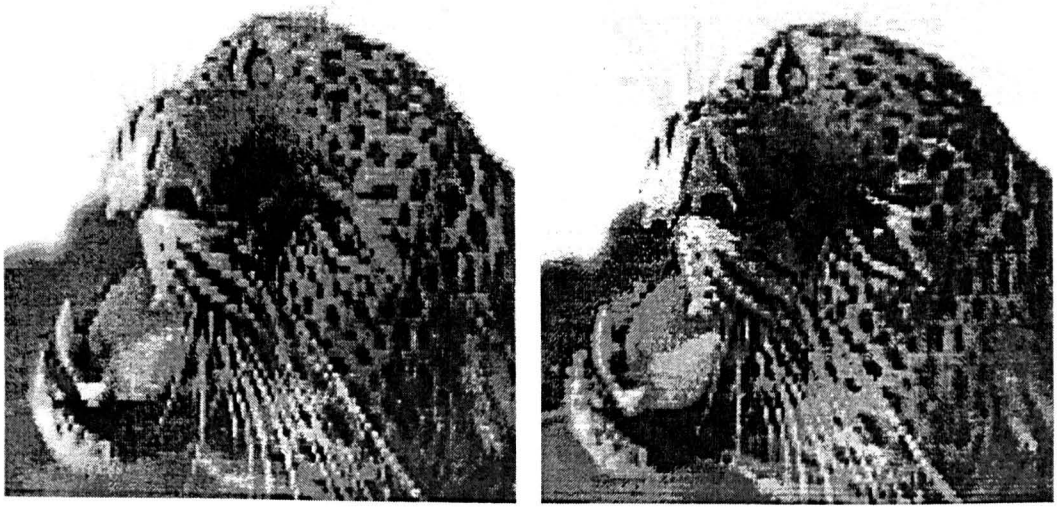
It is generally accepted that wavelet packets give better results.



(a)



(c)



(b) (c)

Figure 7.33. Wavelets versus Wavelets packets

(a) & (b) Use wavelets, (c) & (d) use wavelet packets,
both have a compression ratio of 8.

7.4.10. Filters

All kinds of filters have a certain maximized or minimized criteria. Depending on that criteria other results are achieved. As for the order of the filter, it is obvious that larger order have better results but longer calculation time.

In figure 7.34 (a) a Daub4 is used and in (b) a Daub12 is used.

(a) (b)

Fig.7.34. Daub12 (b) gives more detailed results than Daub4 (a)

7.4.11. Further Research

The intention of this part of my research was to program some principal applications with wavelet transforms on the C80. Only the basic wavelet compression algorithms were realised to find out about the possibilities of the system. More fundamental problems will have to be considered in the future :

- Instead of totally discarding subbands, they could be coded using less bits per pixel. In our experiments we stayed at 8 bpp. Using 4 bpp instead of nothing , would mean a compression ratio of only 2 for certain subbands. The total would be less compressed, but the image would contain more information and would approximate the original image in a better way. This would, of course, include longer information headers.

- The most annoying aspect of wavelet-compressed images are the ripples. Software solutions must be made available to solve it.

- To implement real-time compression, a lot of the C-code can be optimized. E.g. using statements like 'Matrix++' instead of 'Matrix[row][col]', the latter uses more

pointer arithmetic. Also, memory management can be improved by using in-matrix operations.

- The algorithms should be generalised to work with arbitrary image sizes in stead of to powers of two . [68] suggests a method for this.

7.5. General Conclusions

The examined hardware (VSP and C80) doesn't really suffice the high demands of real time video processing.

- As the ALUs in the VSPs are rather primitive constructions, a lot of them are needed to perform for instance Wiener filtering in wavelet space.(par 7.2.6) VSP2 is about 4 times as powerful as VSP1 (see fig.7.1) but doesn't solve the problem that one needs a large amount of ALUs and other building blocks to realise the algorithm in real time. Although the components can be virtually present it becomes very difficult to control their individual efficiency with software tools.
- Philips Research Lab (NATLAB) suggest to solve the problem by considering an ASIC solution with VSP building blocks. It gives of course so more freedom but one stays confronted with huge amounts of primitive ALUs which has to be efficiently mapped on a chip.
- With the TMS320C80 we are confronted with more complex slave processors than the ALUs in the VSP. They can address all the pixels in the picture while the VSP only works on the lines. It is also programmable in C . This should make it easier to program and to simulate the results. However this is a relative expression : The Texas Instruments' 'hot line' in Paris was not able to provide us with sufficient support to let the slave processors work harmoniously together. We are however confident that this problem will be solved very soon.
- The experiments we were able to perform told us that the C80 was not effective enough to realise for instance real time Wiener filtering in wavelet space. New boards with 5 C80s are now coming on the market. Question

arises again, as with the VSPs, how efficiently their slave processors could be programmed!

CHAPTER 8

General Conclusions

When this thesis started the goal was threefold :

- Explore some advanced mathematical fields which were applicable in new electronic devices.
- Study and solve the mathematical problem with computer software tools.
- Implement the results on advanced DSP chips and investigate the architecture for new custom made chip design.

These action points harmonise with the intention of three research groups wanting to collaborate in an ESPRIT project .

- The Computing Sciences department of the De Montfort University is a well established school of mathematicians and Prof. Blackledge's Research Unit is a centre of excellence for image processing .
- The micro-electronics and DSP group from the Katholieke Hogeschool Brugge Oostende which have experience in building chips and boards for DSP and image processing.
- The DSP group from IMEC (Inter university Micro-Electronic Centre) Leuven -Belgium which is involved with the implementation of new schemes and technologies for semi-conductor manufacturers all over the world.

The covered area for the thesis was about the same as the ESPRIT proposal and went from wavelet mathematics to new DSP chip concepts. Partners in industry were found to collaborate in the project if the proposal was accepted..

Summary of the research

An extensive literature study revealed the vast application field of wavelets. Noise reduction and compression techniques were studied and implemented in software.

- For noise reduction the Wiener filtering in wavelet space gave the best results (Chapter 4). Normal Wiener filtering is ideal, granted the original signal or image (without noise) is

known.[59] This is rather unrealistic, therefore working in wavelet space (it provides an elegant way to investigate the variance of the noise at the first level of the wavelet analysis) is preferred. Different publications in Astronomy and Astrophysics [24]-[29] prove that, compared with for instance thresholding , wavelet filtering is a superior technique.

- For the implementation on microprocessors research on the most efficient FWT algorithm was performed.(Chapter 6). Biorthogonal wavelets with the lifting scheme and based on spline functions proved to be excellent for hardware implementations because :
  1. The filter coefficients are rational numbers with denominators being powers of 2. (for example 1/8). The multiplication with these kind of filter coefficients (1/8) can be reduced to a simple '3 shift right' operation on the data which is of course much quicker implemented than a floating point multiplication.
  2. The lifting scheme provides, with the predict and update steps, the possibility to integrate the 'low pass and high pass filtering' in one operation. This resulted in algorithms twice as fast as Mallat's FWT.
- The microprocessors chosen (Chapter 7) for the implementation were selected on the following criterion : They should be parallel processors and capable of real time image processing and they should also show an interesting architecture for further implementation in full custom design. Both the VSP from Philips and the MVP from Texas Instruments were thought to fulfil these needs.

  The most promising algorithms were realised on those processors : the VSP was programmed with the Wiener filter in a wavelet-lifting scheme space. The MVP was programmed with Haar and Daubechies wavelet packets for data compression. The selected criterion was maximum entropy or variance.

  For the **VSP** the results were moderate positive : the algorithm functioned in real time. However, though a huge amount of parallel processors involved, the possible number of levels in the wavelet transform stayed limited to 2. To create state of the art technology for wavelet filtering the VSP architecture had to be changed in at least 4 aspects :

  1. More accuracy : 12 bits, even in double accuracy is not good enough for wavelet analysis at more than 2 levels..
  2. More powerful VSPs, not like VSP2 but with more elabrated ALUs. An Harvard architecture, like on most DSP processors, would be appropriate.
  3. Better software to assist in the equal distribution of the tasks over the involved processors; now it runs too quickly into chaos.

4. Larger memory capacity to memorise the whole screen. Now only screen lines can be memorised and processed.

These remarks were passed to Philips. After some time we were informed that the VSP project was cancelled. Other labs like IMEC for instance wrote also negative reports and we think that together with some financial problems in the company this negative decision was taken.

The results with the **MVP** were also not completely positive. The architecture was, contrary to the VSP, appropriate for the job. Though images could successfully be compressed and decompressed , the maximal speed was not reached because the 4 parallel processors refused to 'collaborate' optimally. The TI advice centre in Paris had to admit that in some applications there were still some software bugs in the parallel operation of the slave processors. Other universities contacted in France and in the UK had the same problems.

Considering this temporary problem and with the limited information of only 1 slave processor running, we suspect that the MVP, in 'optimal' condition, is capable of performing data compression with wavelet packages (2 levels) on 1-4 pictures per second, depending on the size of the image (256×256 or 512×512).

Future semi-custom and full-custom design for MPEG4 will incorporate wavelet compression. Properties for the core processor for this application could be deduced from the former results :

1. A versatile core processor should be build with multiply/accumulate/decimate facilities . The MVP is a good architecture, although a slightly simpler architecture could also satisfy.
2. The ideal scheme should be a compromise between the versatility of the VSP (many ALUs) and the complexity of the MVP (full capacity DSPs).

To conclude, future development and investigation on real time processing will surely need more powerful systems. New PCI boards, with 5 MVPs on it, are becoming available. With some ESPRIT funding, I hope we will be able to continue this collaborative research and complete it.

Mathcad's short reference guide

Mathcad ® 6.0 was chosen because of its ability to **write and execute** formulae in real mathematical notation . Mathcad calls it the ' Live Document Interface'. Variables and formulae are combined into a program and the outcome can be graphically displayed.

Only the instructions to understand the programs in the text will be briefly reviewed.

1. Counters and Variables

A counter of N=20 integer values starting from 0 is edited as follows :

$$N := 20 \qquad n := 0..N-1$$

Note the : = expression to **define** a **local** constant or a counter. Sometimes the ≡ is used to make the variables or functions **global** . (Not dependant on the position on the page).

When the incremental value is different from 1 the variable can also be defined like

$$t := -\pi, -\pi + \frac{\pi}{N}..\pi$$

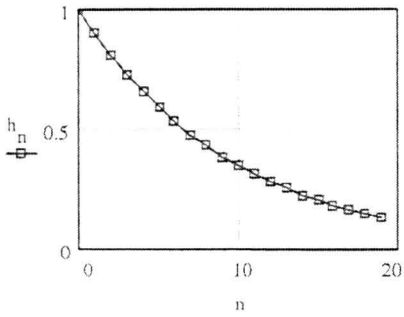
The variable t is now defined between -π and π with increments of π/20. (N=20 as defined above).

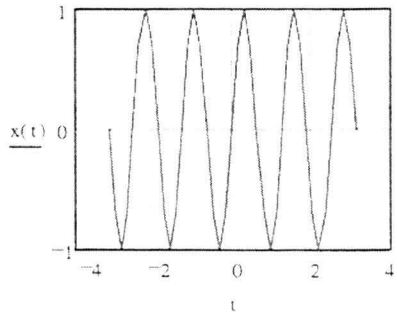
2. Vectors, Matrices and Functions

Counters and variables can be used to produce vectors, matrices and functional results. As an example we generate a decaying function and a sine wave.

$N := 20 \qquad \alpha := .9 \qquad n := 0..N-1 \qquad h_n := \alpha^n$

$t := -\pi, -\pi + \frac{\pi}{N}..\pi \qquad \omega := 5 \qquad x(t) := \sin(\omega \cdot t)$





Vectors **start from index zero** (or one : programmable in Mathcad menu bar) . Functions can take **positive and negative arguments**. The results can be displayed discretely, interconnected or a combination of both.

Matrices are programmed as two-dimensional vectors and generated as :

$N_1 := 4 \quad N_2 := 5 \qquad \alpha := .5 \quad \beta := 2 \qquad i := 1..N_1 \qquad j := 1..N_2$

$M_{i,j} := i^{\alpha} \cdot j^{\beta} \qquad M = \begin{pmatrix} 1 & 4 & 9 & 16 & 25 \\ 1.414 & 5.657 & 12.728 & 22.627 & 35.355 \\ 1.732 & 6.928 & 15.588 & 27.713 & 43.301 \\ 2 & 8 & 18 & 32 & 50 \end{pmatrix}$

Remark the notation of the variables N_1 and N_2 and the matrix $M_{i,j}$. 1 and 2 are just indices of N to indicate the difference between variables; i and j are the column and row counters for the matrix M. Note also that = is used to **evaluate** the matrix.

Submatrices can be defined using the following instructions :

$M_{2313} := \text{submatrix}(M, 2, 3, 1, 3) \qquad M_{2313} = \begin{pmatrix} 1.414 & 5.657 & 12.728 \\ 1.732 & 6.928 & 15.588 \end{pmatrix}$

Only the columns out of matrices can be defined directly as vectors. The transpose operation ^{T} is however available to indirectly, define row vectors as well.

$M_{C2} := M^{<2>}$ $M_{C4} := M^{<4>}$ $M_{R4} := (M^T)^{<4>}$

$M_{C2}^T = (4 \quad 5.657 \quad 6.928 \quad 8)$ $M_{C2}^T = (4 \quad 5.657 \quad 6.928 \quad 8)$ $M_{R4}^T = (2 \quad 8 \quad 18 \quad 32 \quad 50)$

The parallel calculation with vectors is coerced by a $\rightarrow$ above the operation, the sum of the vector elements is defined by a $\sum$ before the vector variable.

$M_{C24} := \overrightarrow{(M_{C2} \cdot M_{C4})}$ $M_{C24}^T = (64 \quad 128 \quad 192 \quad 256)$ $\sum M_{C24} = 640$

3. The if Statement and Defining Piecewise Continuous Functions

if(*cond, tval, fval*) Returns tval if cond is nonzero (true)
Returns fval if cond is zero (false)

Some examples :

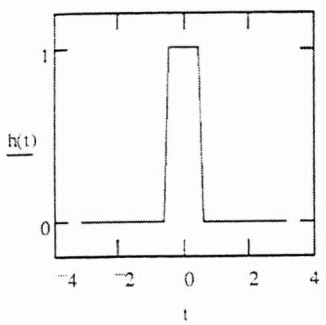
For integers m and n the Kronecker delta function is equivalent to

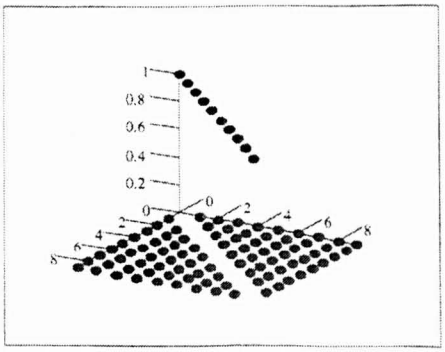
$N := 10$ $m := 0..N-1$ $n := 0..N-1$ $\delta_{m,n} := if(m=n, 1, 0)$

The equal sign in m=n is used in a conditional statement.

A pulse equal to 1 in the interval -.5, .5 and elsewhere 0 is defined by

$h(t) := if(-.5 \le t \le .5, 1, 0)$





The Kronecker delta function is drawn as a 3D bar chart with maximum spacing specifications between bars.

4. Symbolic Calculations

In the symbolic mathematics another equal sign (→) is used :

$$\begin{pmatrix}1\\1\\1\\0\\0\end{pmatrix}^{T} \cdot \begin{pmatrix}1\\\alpha\\\alpha^2\\\alpha^3\\\alpha^4\end{pmatrix} \rightarrow 1 - \alpha - \alpha^2$$

Note that the first vector is transposed to make a correct matrix multiplication.

5. Programming

New functions can be defined by programming Mathcad instructions in a sequential order. The ← is typical for partial results in the program. As an example, we consider Newton's iteration process to calculate the intersection of 2 curves and then compare the result with the **given-find** technique from Mathcad.

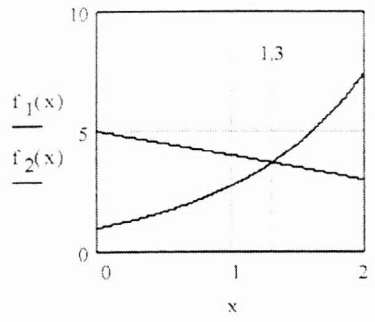
$f_1(x) := e^x$ $\quad f_2(x) := x - 5$ $\quad$ define : $f(x) = f_1(x) - f_2(x)$

$f(x) := e^x - x - 5$ $\quad \frac{d}{dx}\left(e^x - x - 5\right) = e^x + 1$ will be defined as dif_f(x)

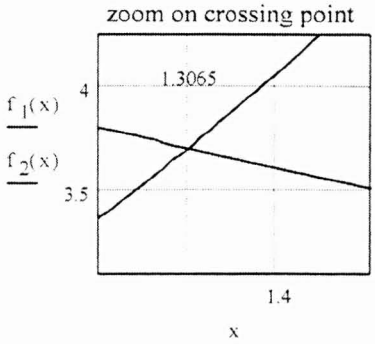
$TOL = 10^{-6}$

$$\begin{aligned}&N\_iteration(f, dif\_f, start\_value) := \\ &\quad first\_new\_value \leftarrow start\_value - \frac{f(start\_value)}{dif\_f(start\_value)} \\ &\quad new\_value \leftarrow first\_new\_value \\ &\quad old\_value \leftarrow start\_value \\ &\quad \text{while } |new\_value - old\_value| > TOL \\ &\qquad new\_value \leftarrow old\_value - \frac{f(old\_value)}{dif\_f(old\_value)} \\ &\qquad old\_value \leftarrow new\_value \\ &\qquad new\_value \leftarrow old\_value - \frac{f(old\_value)}{dif\_f(old\_value)} \\ &\quad new\_value\end{aligned}$$

N_iteration(f, dif_f, 1) = 1.30655864



zoom on crossing point



given

$e^x + x - 5 = 0$

x := 1 : starting point

x_crossing := find(x)

x_crossing = 1.30655864

6. Digital Signal Processing

The FFT and IFFT algorithm is implemented with **fft**(x) and **ifft**(x) for x having $N=2^L$ elements. ($L \in \mathbf{Z}$). When $N \neq 2^L$ **cfft**(x) and **icfft**(x) are used. (See examples section 2.13). In the forward and backward transform a constant $1/\sqrt{N}$ before the Σ sign in the FFT formula is used this in apparent contradiction with the definition in Eq. 2.14. Both definitions are however correct.[56],[63]

A DSP toolkit is used to perform expressions like :

response(x,A,N) : Response of a filter A to an input signal x over a N element interval.(See example section 4.5.5)

resample(x,m,n) : Decimation of x by m. (See example section 4.5.5)

corr(x,y) : Returns the correlation coefficient between x and y.

gaussn(N) : Returns N random numbers with mean = 0 and variance =1.

For wavelets only **dwavelet(x)** and **idwavelet**(x) are available for Daubechies 4 on 2^L elements. (See example section 4.55)

APPENDIX 5A

Introduction

I would like to report in this appendix on how I became acquainted with wavelets. The idea of the existence of oscillatory signals which could be used in inner products to produce sets of orthogonal coefficients intrigued me. Being an engineer I became very much interested in the filter bank idea. So I designed a FIR filter whose impulse response was almost of a compact support (died out very quickly).

I then wrote the forward and inverse wavelet transform and started my experiments. All these preliminary assays, although afterwards of not so much use, taught me the essence of wavelets. I quickly became aware of the difficulty to produce errorless forward and backward transforms. Only very special sets of filter coefficients did the job perfectly! The Haar filter coefficients were the first I rediscovered! While I was searching for other coefficients I fine tuned my software for the transforms. Finding new wavelets by using filter banks proved to be a hard job and I had to admit that a more abstract approach was necessary. Just like Morlet who went to Grossmann I had to consult Daubechies [57] to fully understand how it should be done. In Wickerhausen's book [67] I found the filter coefficients and gradually I became aware how to design them (Appendix 5B,6A). Later on my interest diverged more in the direction of application than on the design of new wavelets.

FIR filter design

The following lowpass characteristic delivers, after IFFT, a very short impulse response. A closed form solution resulted in :

$$h(n) = \frac{1}{n\pi(1-4\pi^2\frac{f_t^2}{f_s^2})}\sin(n\pi\frac{f_a+f_p}{f_s})\cos(n\pi\frac{f_t}{f_s})$$

f_a and f_p are the frequencies where the roll off starts and ends. In our example we choose : $f_p = \pi/6$ and $f_a = 5.\pi/6$. $f_t = f_a - f_p$ f_s = sampling frequency.

$N := 100$

$$\Omega := 0,2\cdot\frac{\pi}{100}\,..\,2\cdot\pi$$

$$H(\Omega) := \begin{cases} 1 \text{ if } 0 \le \Omega \le \frac{\pi}{6} \\ \left[1+\cos\left[\frac{3\cdot\left(\Omega-\frac{\pi}{6}\right)}{2}\right]\right]\Big/2 \text{ if } \frac{\pi}{6} < \Omega \le 5\cdot\frac{\pi}{6} \\ 0 \text{ if } 5\cdot\frac{\pi}{6} < \Omega \le 7\cdot\frac{\pi}{6} \\ \left[1+\cos\left[\frac{3\cdot\left(\Omega-\frac{3\cdot\pi}{6}\right)}{2}\right]\right]\Big/2 \text{ if } 7\cdot\frac{\pi}{6} < \Omega \le 11\cdot\frac{\pi}{6} \\ 1 \text{ if } 11\cdot\frac{\pi}{6} < \Omega \le 2\cdot\pi \end{cases}$$

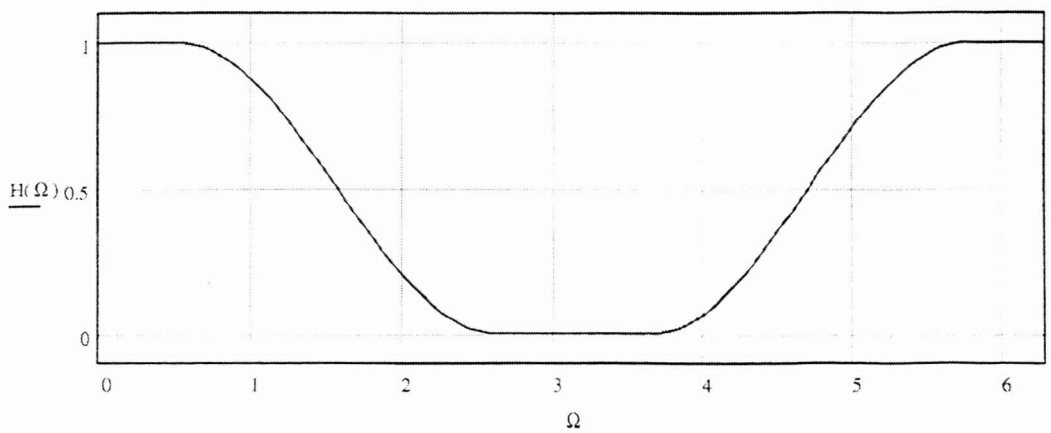


Fig.5ap.1. Spectrum of low pass prototype FIR filter.

$$H1_k := H\left(k \cdot 2 \cdot \frac{\pi}{100}\right) \qquad h1 := \frac{1}{\sqrt{N}} \cdot icfft(H1)$$

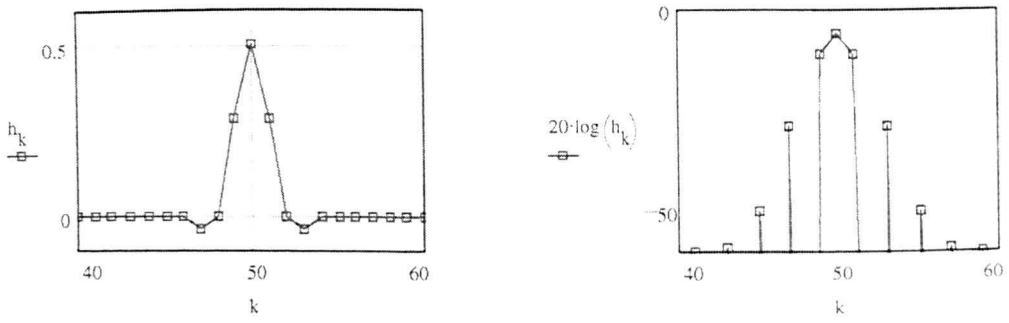


Fig.5ap.2. The corresponding impulse response very quickly decays to zero. Within 20 filter coefficients the amplitude is decremented by 60 dB.

These filter coefficients are used for the low pass prototype. The accompanying high pass filter has coefficients defined by :

$$n := 0..20$$

$$h_{prototype_n} := h_{n+40} \qquad g_{prototype_n} := (-1)^n \cdot h_{prototype_n}$$

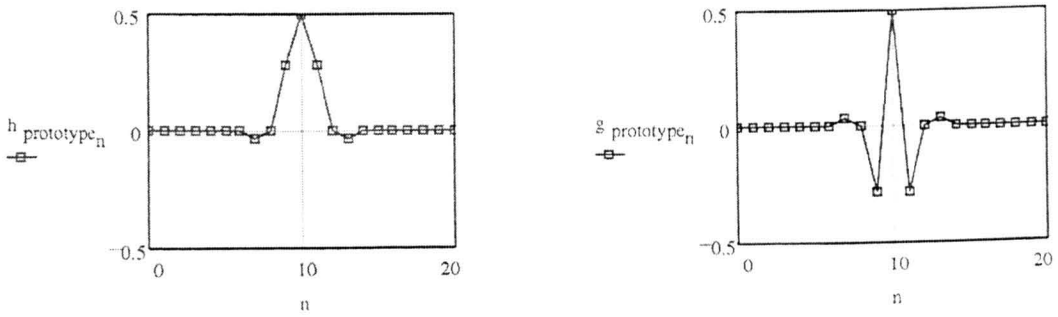


Fig.5ap.3. Low pass and accompanying high pass FIR filter coefficients.

The spectral amplitude of the FIR filters shows good resemblance with the original prototype (fig.5ap1.)

$h_{ext_k} := 0$ $h_{ext_n} := h_{prototype_n}$ $H_{prototype} := \sqrt{N} \cdot cfft(h_{ext})$

$g_{ext_k} := 0$ $g_{ext_n} := g_{prototype_n}$ $G_{prototype} := \sqrt{N} \cdot cfft(g_{ext})$

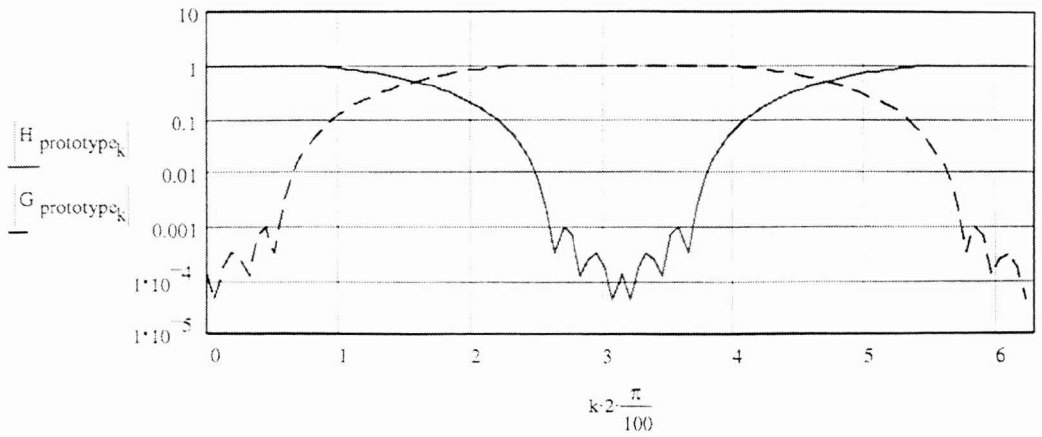


Fig.5ap.4. Logarithmic characteristic of amplitude spectrum of lowpass and highpass prototypes.

Wavelet design

Two way were considered to improve the performances of the filters :

- Change the amplitude spectrum in such a way that the impulse response obtained via IFFT is even more compact support.
- Change the filter from linear phase to minimal phase.

The first approach resulted in the rediscovery of the Haar wavelet.

$N := 101$

$\Omega := 0, 2 \cdot \frac{\pi}{100} .. 2 \cdot \pi$

$$H(\Omega) := \begin{cases} 1 \text{ if } 0 \le \Omega \le \frac{3 \cdot \pi}{12} \\ \left[1 - \cos\left[\frac{3 \cdot \left(\Omega - \frac{3 \cdot \pi}{12} \right)}{2} \right] \right] \text{ if } \frac{3 \cdot \pi}{12} < \Omega \le 11 \cdot \frac{\pi}{12} \\ 0 \text{ if } 11 \cdot \frac{\pi}{12} < \Omega \le 13 \cdot \frac{\pi}{12} \\ \left[1 + \cos\left[\frac{3 \cdot \left(\Omega - \frac{5 \cdot \pi}{12} \right)}{2} \right] \right] \text{ if } 13 \cdot \frac{\pi}{12} < \Omega \le 21 \cdot \frac{\pi}{12} \\ 1 \text{ if } 21 \cdot \frac{\pi}{12} < \Omega \le 2 \cdot \pi \end{cases}$$

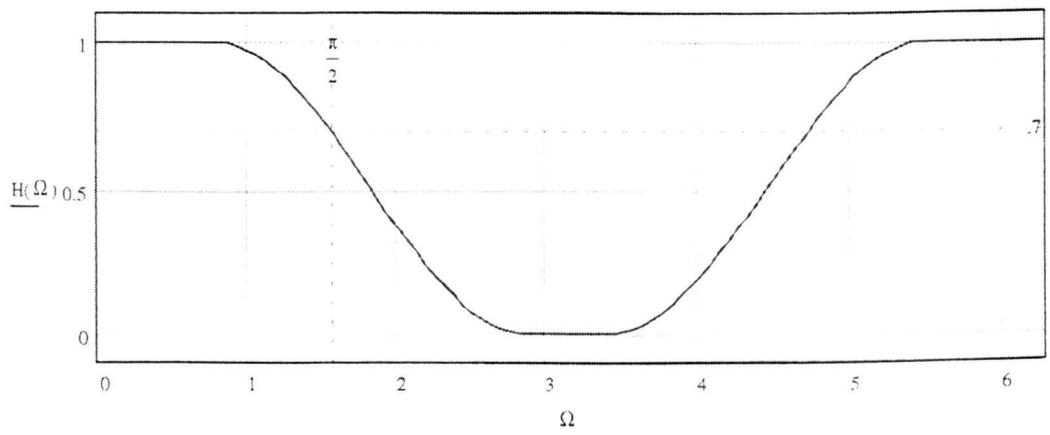


Fig.5ap.5. The pass band of the low pass prototype has been changed, compared with the first attempt the attenuation at π/2 is now only 3dB.

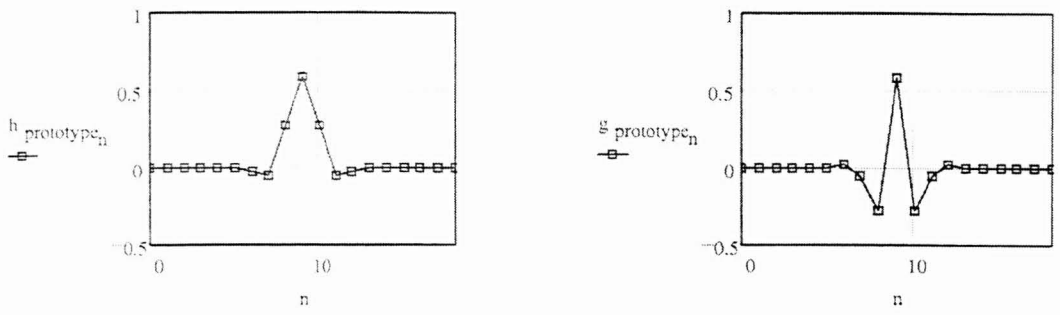


Fig.5ap.6. The impulse response for the low pass and high pass filter characteristic of fig.5a.7.

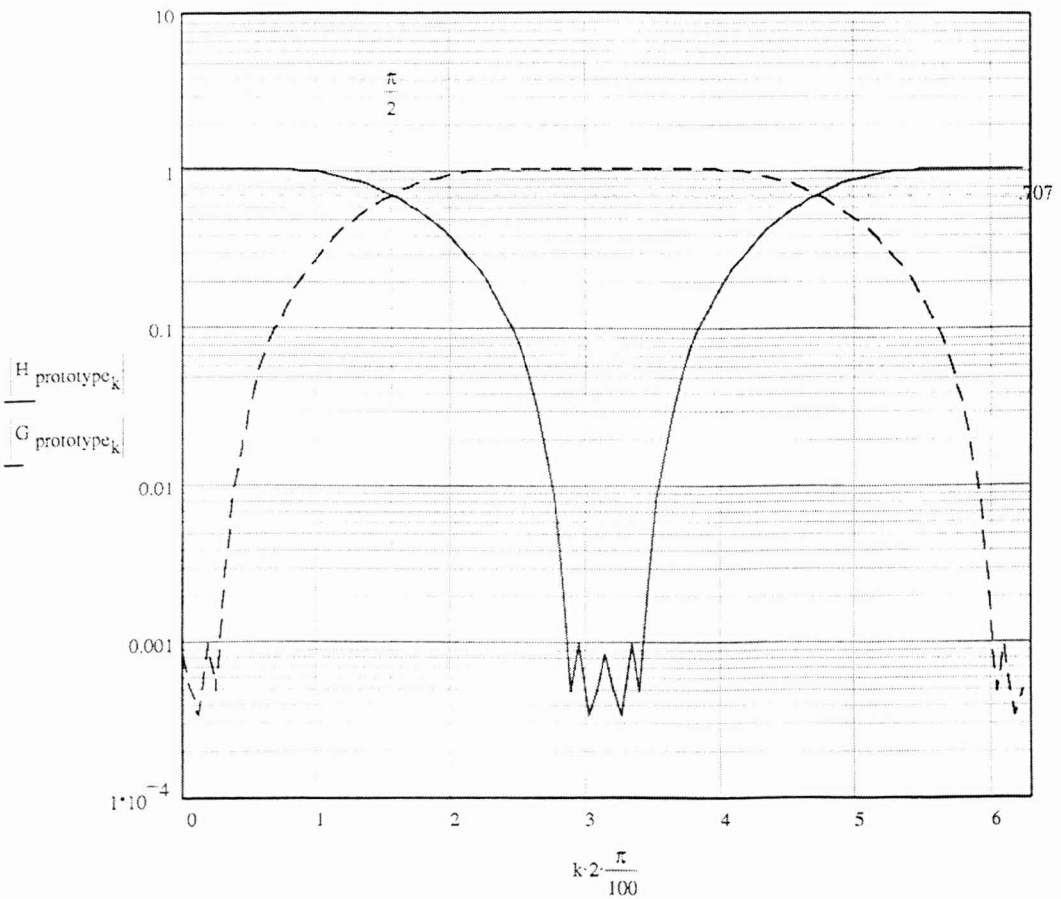


Fig.5ap.7. The matching of the characteristics is perfect but the attenuation not yet good enough.

The solution with an even number of filter coefficients finally resulted in an almost perfect wavelet. Fig.5ap.8. shows the Haar filter coefficients as a result of inverse Fouler transforming!

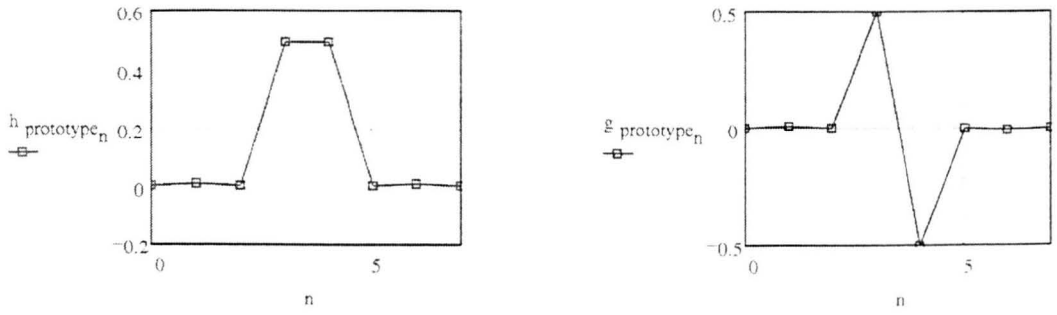


Fig.5ap.8. A good approach of the Haar wavelet filter coefficients

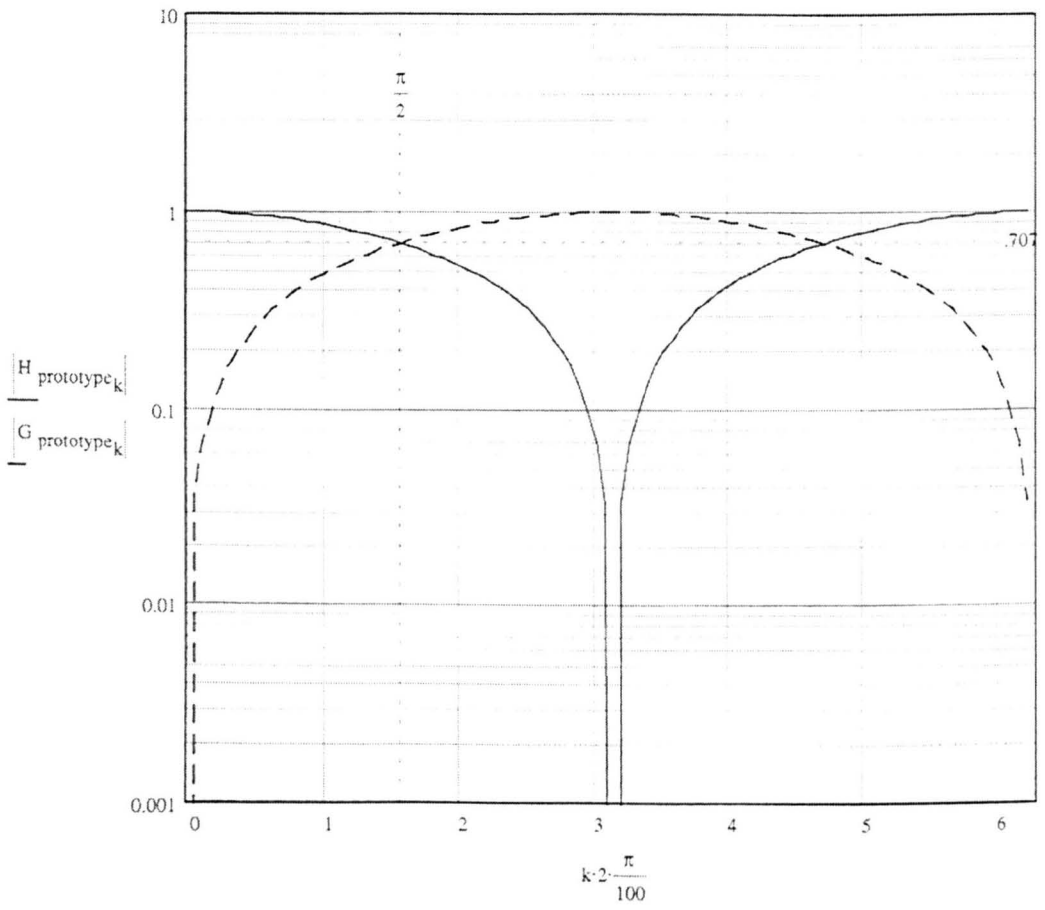


Fig.5ap.9. Remark the almost similar spectral characteristic then the one from fig. 5a.7. The attenuation at π goes however much deeper.

We now analyse the 'almost' Haar wavelet and discover that the smallest errors result in rather big changes in the reconstructed signal.

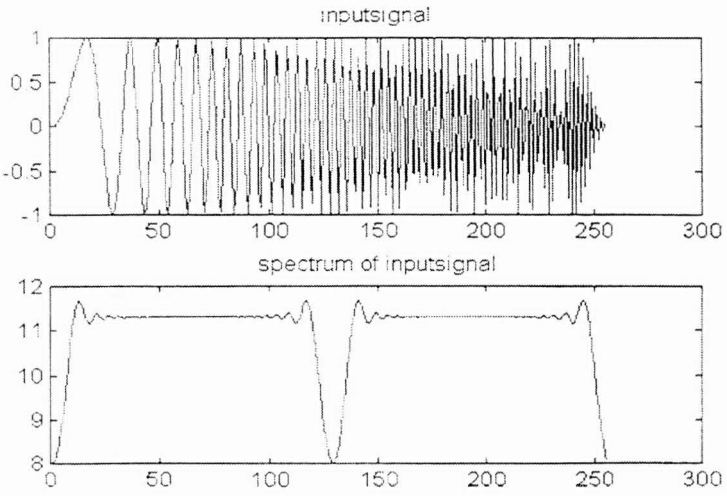


Fig.5ap.10. The chirp signal and its spectrum as input.

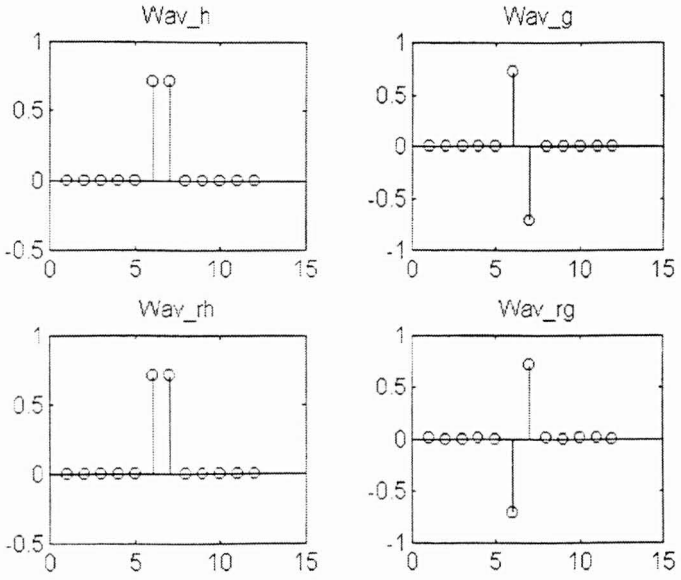


Fig.5ap.11. The wavelet filter coefficients

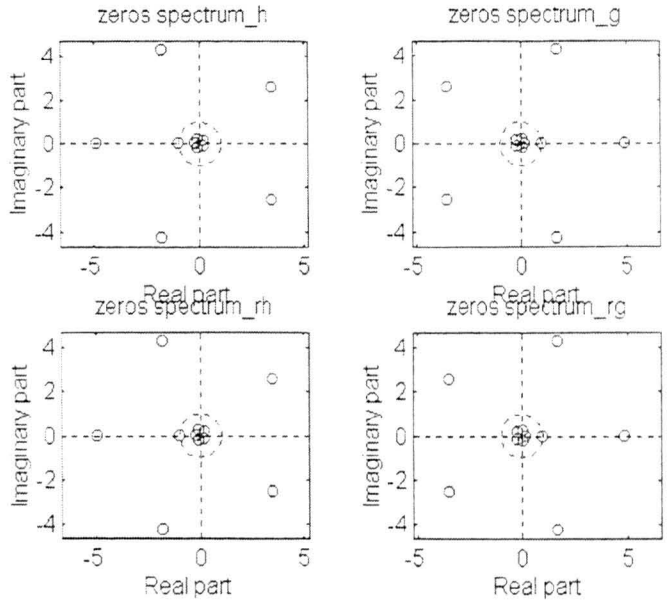


Fig.5ap.12. Zeros of wavelet filter

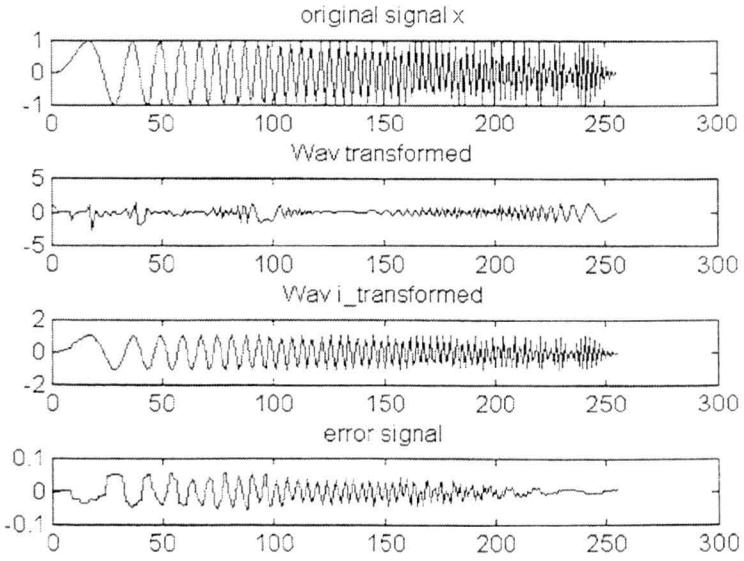


Fig5ap.13. The analysed and reconstructed chirp signal.

Fig.5ap.14. Remark the error in the spectrum of the reconstructed signal.

To conclude, wavelet filter coefficients must be very precisely defined otherwise the errors increase dramatically. I refer to Daubechies' 10 Lectures on Wavelets' for a more detailed study on smoothness and differentiability of wavelets.

At the beginning of the experiments the importance of vanishing moments was not very clear therefore the phase of the filter responses was wrongly blamed for the error. So, phase modifications were investigated to modify the filters in a phase minimal sense. This resulted in asymmetrical impulse responses.. It was accomplished by moving the position of the zeros of the filters in the z-plane. **Phase linear** systems have zeros which are not only complex conjugate, they also appear in couples with the same angle but with reciprocal amplitudes compared with the unit circle. (Fig 5ap.18). A **phase minimal** system was made of it by removing the reciprocity and putting all the poles inside the unit circle. (Fig5ap.18.).

This resulted in some nicely shaped asymmetric filter coefficients but the results in the transform were, as later understood, disappointing. The following figures report about this design evolution. Compared with real wavelets it became rather close but not good enough. Wickerhausen reports [67] in this context about a wavelet designed by Vaidyanathan on an experimental basis. Daubechies said also that Meyer's first wavelet originated almost in a miraculous way! However once the algorithms were

understood and the filter coefficients known, the application of exiting wavelets became more important than the design new ones. So this was left behind.

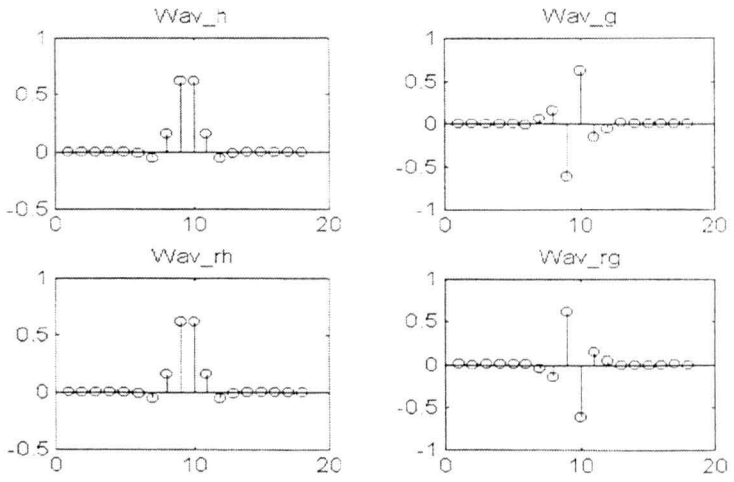


Fig.5ap.15. The impulse responses : wav_h and wav_g for the forward transform and wav_rh and wav_rg for the backward transform.

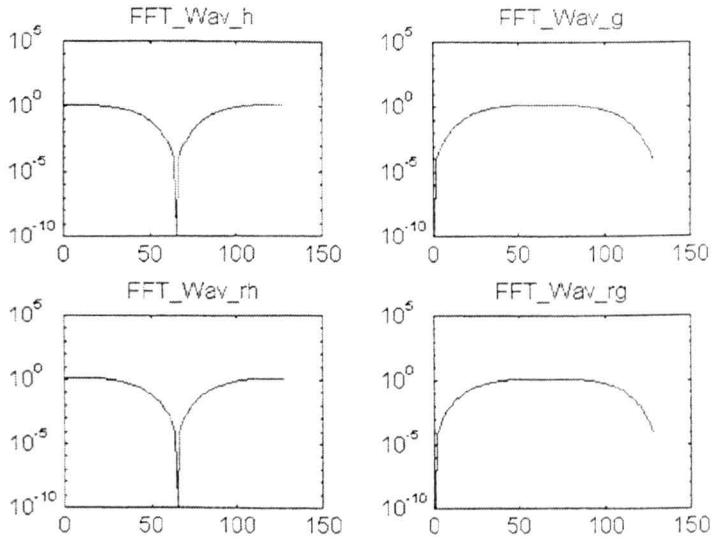


Fig.5ap.16. The spectra show excellent attenuation at half sampling frequency.

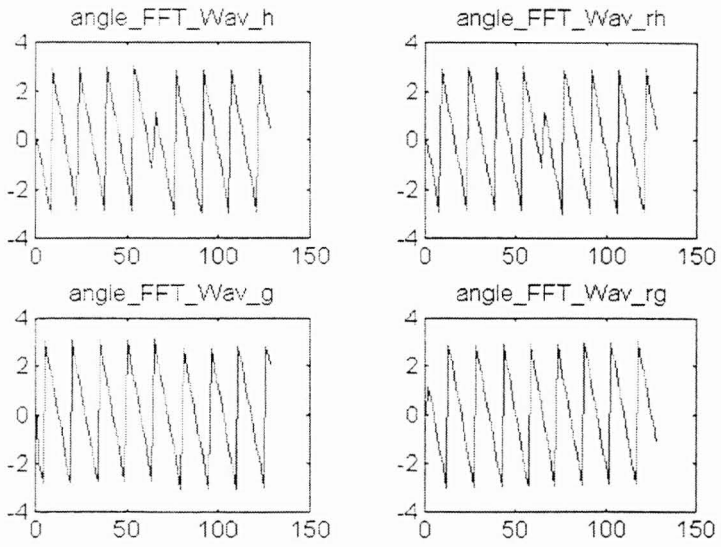


Fig.5ap.17. The phase response is linear for all filter characteristics.

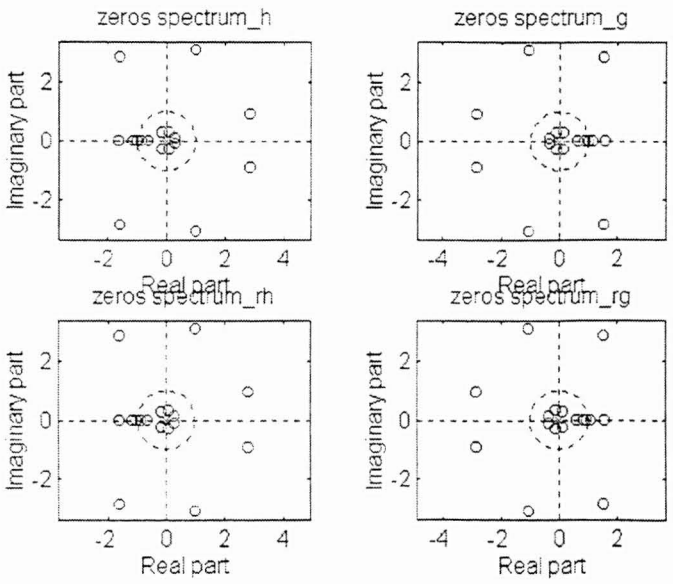


Fig.5ap.18. The roots (zeros) of the z-transform of the different impulse responses can be visualized in the z plane. These are all phase linear filters because their zero pairs are reciprocal against the unit circle.

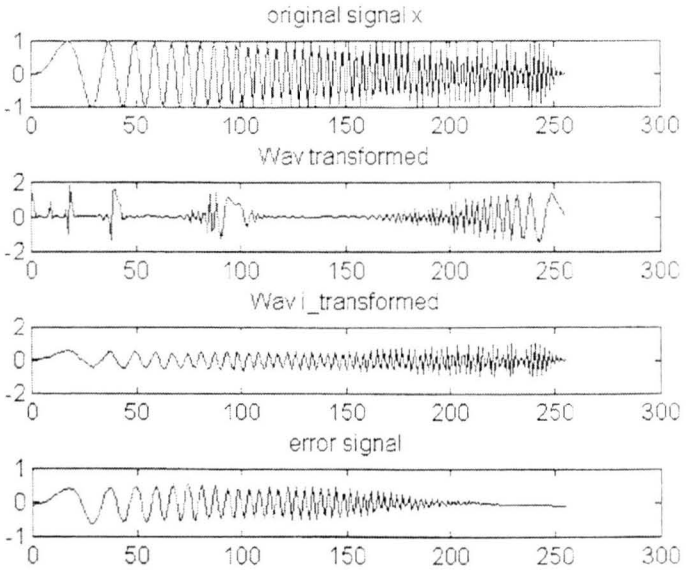


Fig.5ap.19. Level 1 gives good results but gradually the error increases. One could initially think that the phase response of the filters is responsible for it.

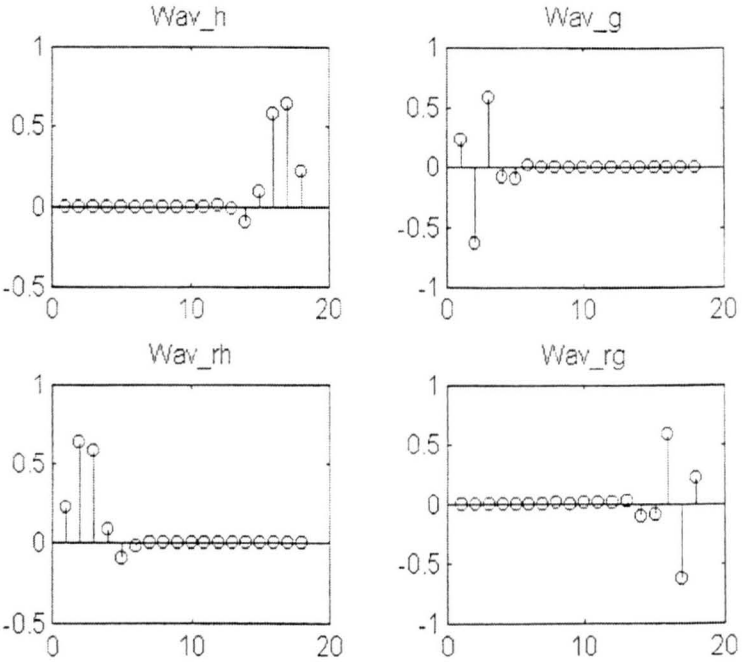


Fig5ap.20. Asymmetrical impulse responses after minimalization of the phase of the symmetrical filters

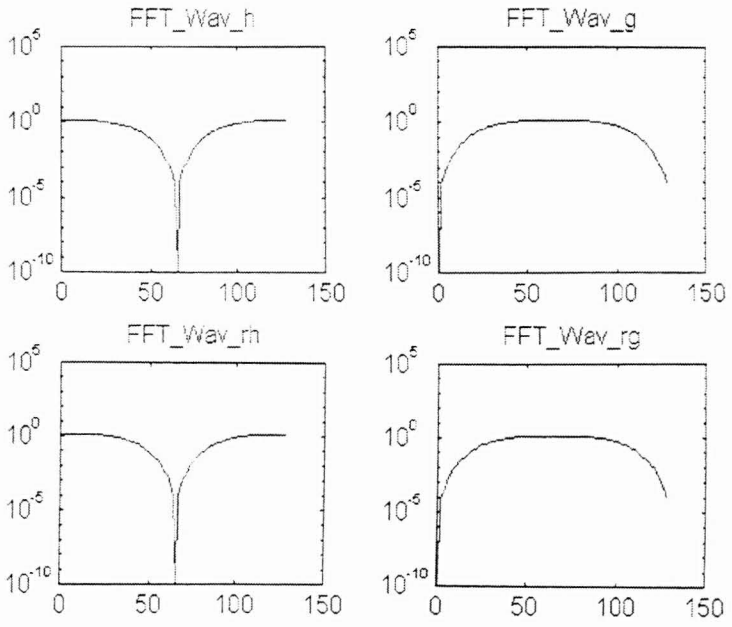


Fig5ap.21. Amplitude components of the spectra of the asymmetrical low and high pass filter coefficients.

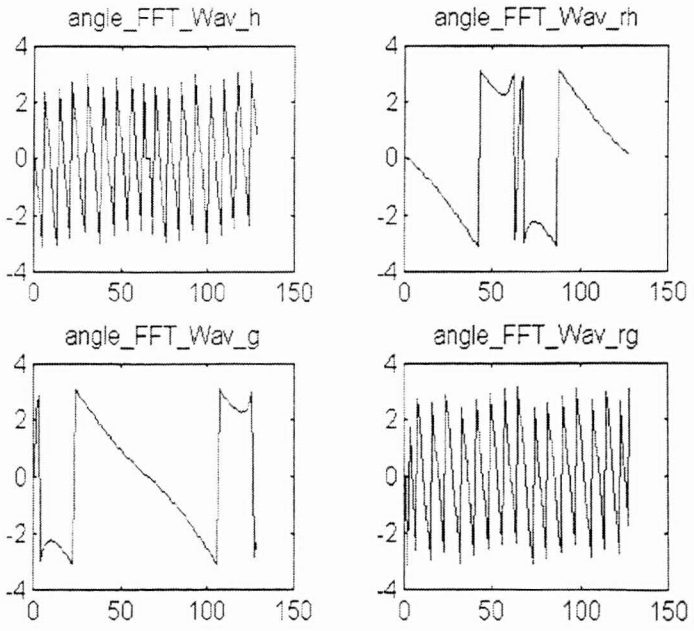


Fig5ap.22. Phase components of the spectra of the asymmetrical low and high pass filter coefficients.

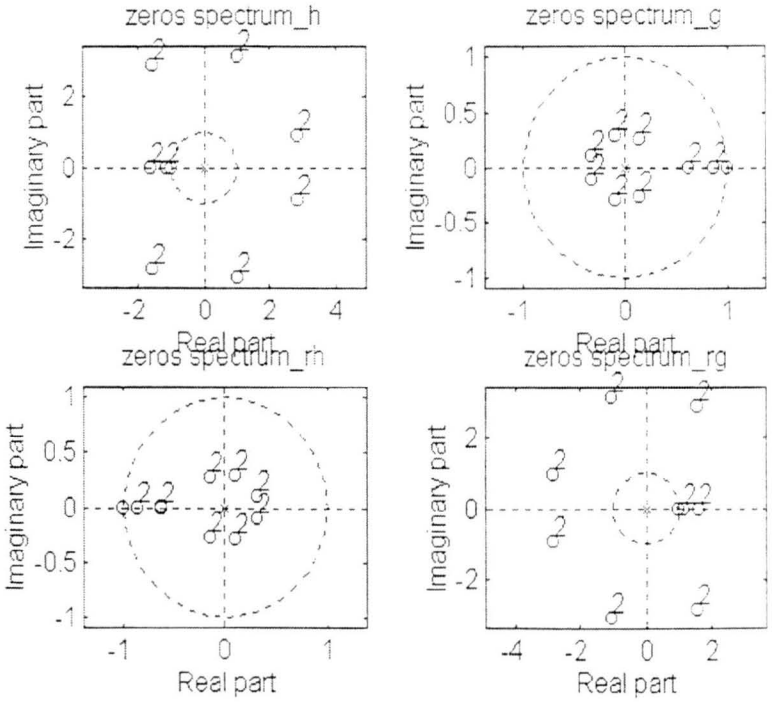


Fig5ap.23. Transfer function zeros of the asymmetrical low and high pass filter coefficients.

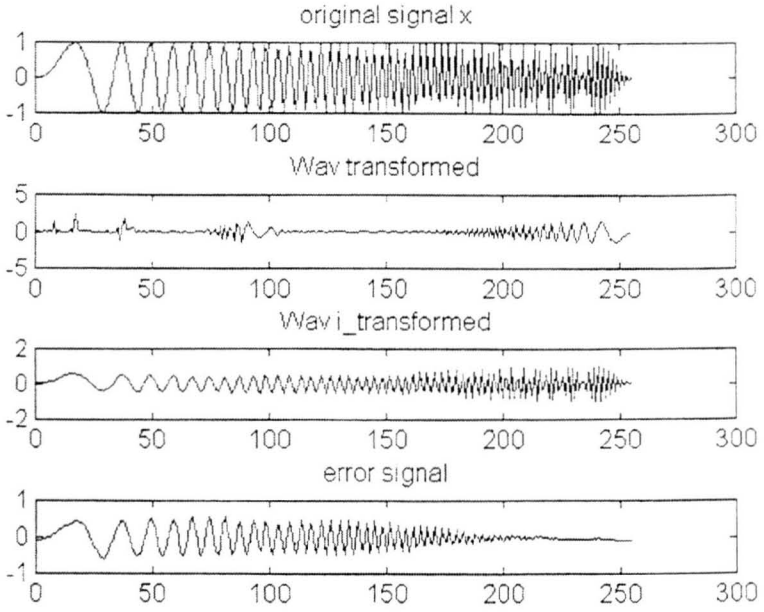


Fig5ap.24. Forward and backward transform using minimal phase filters resulting in rather significant errors at higher levels in the transform.

Coifman Wavelets

Compared with Coifman 18 the wavelet filter coefficients look rather similar to the ones designed above but vanishing moments were considered. By doing so, it results in errorless analysis/synthesis.(Appendix 5b)

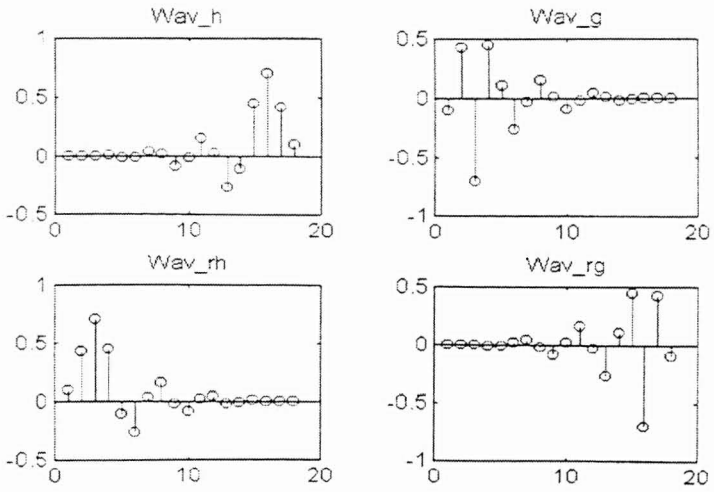


Fig5ap.25. Coifman 18 filter coefficient set.

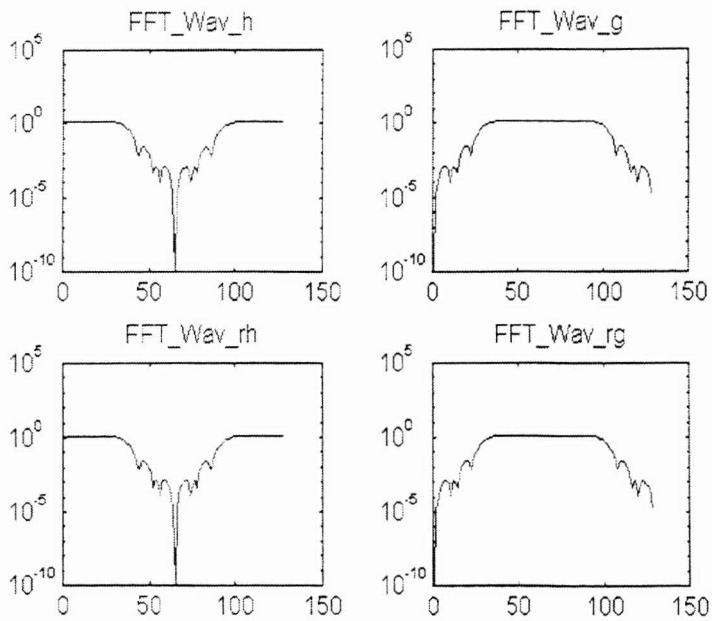


Fig5ap.26. Amplitude components of the spectra of the C18 low and high pass filter coefficients.

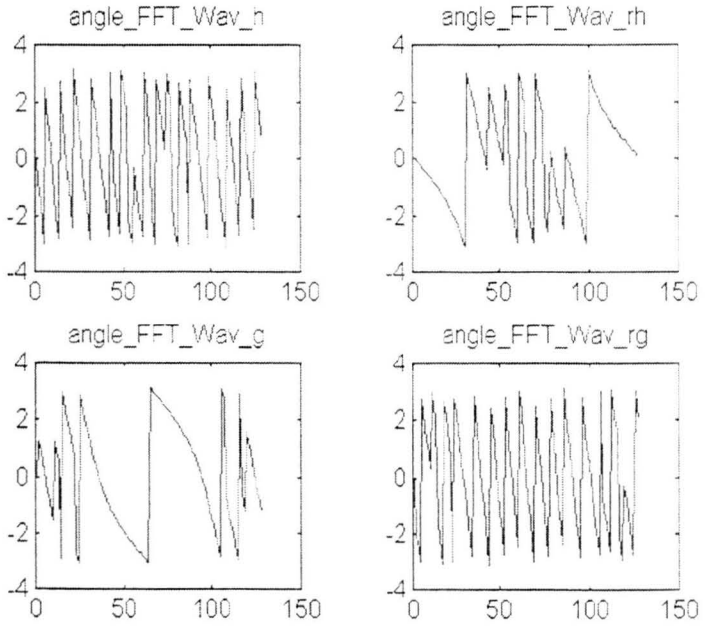


Fig5ap.27. Phase components of the spectra of the C18 low and high pass filter coefficients.

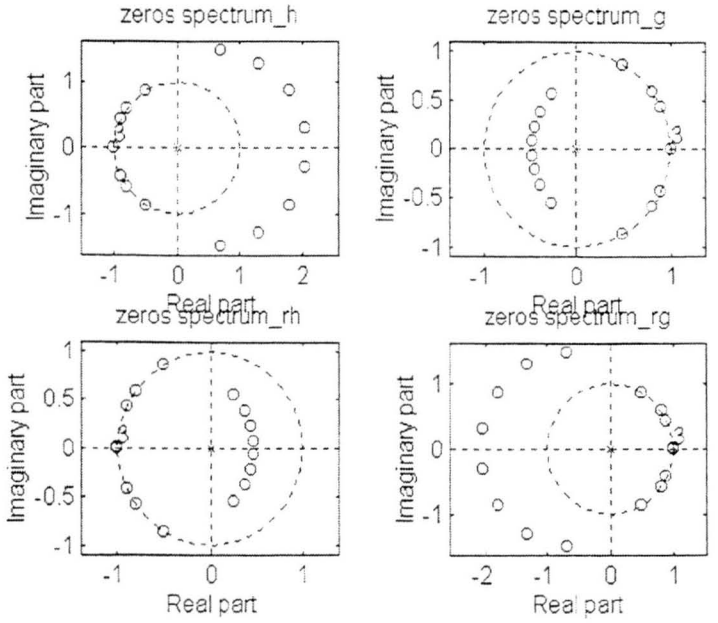


Fig.5ap.28. The roots (zeros) of the z-transform of the C18 Coifman filter coefficients.

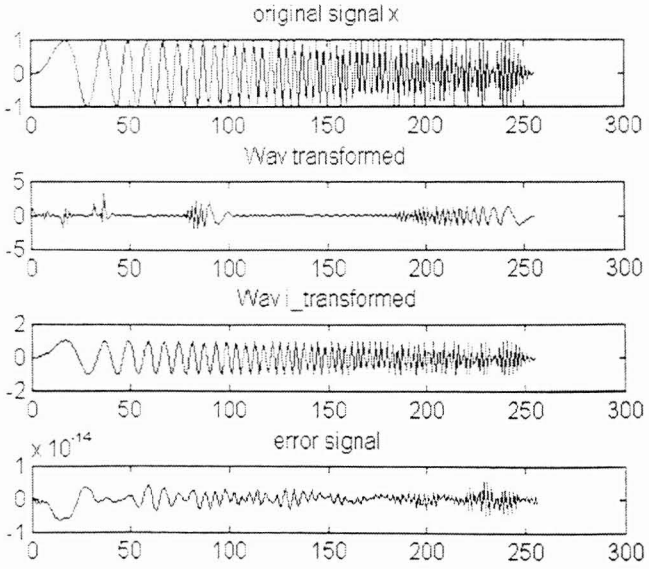


Fig5ap.29. Forward and backward transform using C18 resulting in almost zero error when the transformed signal is subtracted from the original one..

APPENDIX 5B

Designing Daubechies wavelets

Daubechies wavelet filter coefficients can be generated by putting some constraints on the scaling and wavelet filter coefficient matrix and on the smoothness of its wavelet filter coefficients [68] :

Orthogonality condition results in : $M_{n,n}^{T} \cdot M_{n,n} = I$

The analysis matrix is defined by :

$$M_{n,n} = \begin{pmatrix} h_0 & h_1 & h_2 & h_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_0 & h_1 & h_2 & h_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_0 & h_1 & h_2 & h_3 \\ h_2 & h_3 & 0 & 0 & 0 & 0 & h_0 & h_1 \\ h_3 & h_2 & h_1 & h_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_3 & h_2 & h_1 & h_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_3 & h_2 & h_1 & h_0 \\ h_1 & h_0 & 0 & 0 & 0 & 0 & h_3 & h_2 \end{pmatrix}$$

The upper part of the matrix contains the scaling filter coefficients, the lower part contains the wavelet filter coefficients, deduced from their scaling version.
The smoothness of the wavelet result in :

$$\sum_{n=0}^{N-1} n^p \cdot g_n = 0 \qquad \text{with} \quad p = 0..P-1$$

To deduce Daubechies4 we need 4 conditions : 2 are coming from the matrix product, the other 2 from the moments. As starting condition we choose an 'extension' of the Haar scaling coefficients.

$h_0 := \frac{1}{4}$ $h_1 := \frac{1}{4}$ $h_2 := \frac{1}{4}$ $h_3 := \frac{1}{4}$

Given

$$h_0^2 + h_1^2 + h_2^2 + h_3^2 = .5$$
$$h_0 \cdot h_2 + h_1 \cdot h_3 = 0$$
$$-h_0 + h_1 - h_2 + h_3 = 0$$
$$3 \cdot h_0 - 2 \cdot h_1 + h_2 = 0$$

$v_{Daub4} := \text{Find}(h_0, h_1, h_2, h_3)$ $\sqrt{2} \cdot v_{Daub4} = \begin{bmatrix} 0.483 \\ 0.837 \\ 0.224 \\ -0.129 \end{bmatrix}$ $Daub4 := \sqrt{2} \cdot \begin{bmatrix} \frac{1+\sqrt{3}}{8} \\ \frac{3+\sqrt{3}}{8} \\ \frac{3-\sqrt{3}}{8} \\ \frac{1-\sqrt{3}}{8} \end{bmatrix}$ $Daub4 = \begin{bmatrix} 0.483 \\ 0.837 \\ 0.224 \\ -0.129 \end{bmatrix}$

For Daubechies8 8 equations are needed : 4 are coming from the identity matrix conditions. The other 4 are coming from the moments.

$h_0 := \frac{1}{8}$ $h_1 := \frac{1}{8}$ $h_2 := \frac{1}{8}$ $h_3 := \frac{1}{8}$ $h_4 := \frac{1}{8}$ $h_5 := \frac{1}{8}$

Given

$$h_0^2 + h_1^2 + h_2^2 + h_3^2 + h_4^2 + h_5^2 + h_6^2 + h_7^2 = .5$$
$$h_0 \cdot h_2 + h_1 \cdot h_3 + h_2 \cdot h_4 + h_3 \cdot h_5 + h_4 \cdot h_6 + h_5 \cdot h_7 = 0$$
$$h_0 \cdot h_4 + h_1 \cdot h_5 + h_2 \cdot h_6 + h_3 \cdot h_7 = 0$$
$$h_0 \cdot h_6 + h_1 \cdot h_7 = 0$$
$$-h_0 - h_1 - h_2 + h_3 - h_4 + h_5 - h_6 - h_7 = 0$$
$$7 \cdot h_0 + 6 \cdot h_1 - 5 \cdot h_2 + 4 \cdot h_3 - 3 \cdot h_4 + 2 \cdot h_5 - h_6 = 0$$
$$7^2 \cdot h_0 + 6^2 \cdot h_1 - 5^2 \cdot h_2 + 4^2 \cdot h_3 - 3^2 \cdot h_4 + 2^2 \cdot h_5 - h_6 = 0$$
$$7^3 \cdot h_0 + 6^3 \cdot h_1 - 5^3 \cdot h_2 + 4^3 \cdot h_3 - 3^3 \cdot h_4 + 2^3 \cdot h_5 - h_6 = 0$$

$$v_{Daub8} := Find(h_0, h_1, h_2, h_3, h_4, h_5, h_6, h_7) \qquad \sqrt{2} \cdot v_{Daub8} = \begin{pmatrix} 0.23 \\ 0.715 \\ 0.631 \\ -0.028 \\ -0.187 \\ 0.031 \\ 0.033 \\ -0.011 \end{pmatrix}$$

Deduction of the wavelet filter coefficients out of the scaling ones :

$n_4 := 0..3$

$minus\_4_{n_4} := (-1)^{n_4}$

$v_{wDaub4} := \overrightarrow{(minus\_4 \cdot reverse(v_{Daub4}))}$

$h_{Daub4} := \sqrt{2} \cdot v_{Daub4}$

$g_{Daub4} := \sqrt{2} \cdot v_{wDaub4}$

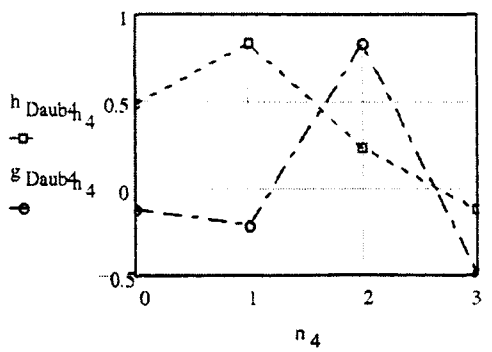
$n_8 := 0..7$

$minus\_8_{n_8} := (-1)^{n_8}$

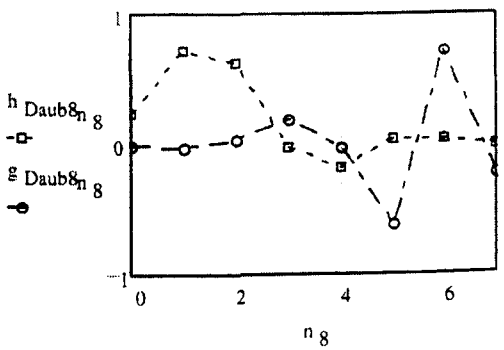
$v_{wDaub8} := \overrightarrow{(minus\_8 \cdot reverse(v_{Daub8}))}$

$h_{Daub8} := \sqrt{2} \cdot v_{Daub8}$

$g_{Daub8} := \sqrt{2} \cdot v_{wDaub8}$



Daub4 : h and g filter coefficients



Daub8 : h and g filter coefficients

For all the other Daubechies wavelets the same procedure can be followed.

Coifman wavelets

Coifman wavelet filter coefficients are generated in an almost similar way than the Daubechies ones. They are designed so that both the scaling function and the wavelet will have vanishing moments. For Coifman6, for the last equation of the Daub6 system with the highest vanishing moment for the wavelet is interchanged with the 'highest vanishing moment' for the scaling function.

$h_0 := \frac{1}{6}$ $h_1 := \frac{1}{6}$ $h_2 := \frac{1}{6}$ $h_3 := \frac{1}{6}$ $h_4 := \frac{1}{6}$ $h_5 := \frac{1}{6}$

Given

$$h_0^2 + h_1^2 + h_2^2 + h_3^2 + h_4^2 + h_5^2 = .5$$

$$h_0 \cdot h_2 + h_1 \cdot h_3 + h_2 \cdot h_4 + h_3 \cdot h_5 = 0$$

$$h_0 \cdot h_4 + h_1 \cdot h_5 = 0$$

$$h_0 - h_1 + h_2 - h_3 + h_4 - h_5 = 0$$

$$-5 \cdot h_0 + 4 \cdot h_1 - 3 \cdot h_2 + 2 \cdot h_3 - h_4 = 0$$

$$5 \cdot h_5 - 4 \cdot h_4 + 3 \cdot h_3 - 2 \cdot h_2 + h_1 = 1$$

$v_{Coif6} := \text{Find}(h_0, h_1, h_2, h_3, h_4, h_5)$ $\sqrt{2} \cdot v_{Coif6} = \begin{bmatrix} 0.227 \\ 0.746 \\ 0.607 \\ -0.077 \\ -0.127 \\ 0.039 \end{bmatrix}$

Compared with the Daub6 this Coif6 has one wavelet vanishing moment less. This results also in 1 zero less at $\Omega = \pi$. (2 instead of 3 roots at -1).

$$\text{polyroots}\left(v_{Coif6}\right)^T = (-1 \quad -1 \quad -0.546 \quad 2.918 - 1.5i \quad 2.918 + 1.5i)$$

APPENDIX 6A

From binomial to B-spline filters

The frequency spectrum of the Haar scaling function has a cosine shape (Appendix 5a). The Haar scaling function has exactly the same characteristics as the B-spline of 0^{th} order. B-splines of the n^{th} order are constructed by n+1 repeated convolutions of the Haar wavelet with itself .

$$\beta^n(t) = \beta^0(t) * \beta^0(t) * \ldots\ldots * \beta^0(t) \quad (n+1 \text{ times})$$

The low pass filter coefficients associated with these B-splines are generated by a binomial serie development.

$$h^n(k) = 1/2^n C^n_i \quad \text{with } C^n_i = n!/(n-i)!i! \; ; i = 0\ldots n$$

The roots of the z transform of these binomial filter coefficients are all located at z = -1. They are of particular interest because zeros at z = -1 are related to vanishing moments. So a binomial filter allied to a B-spline of the 2^{nd} order has 2 vanishing moments. Next step will be to find the dual set of filter coefficients ($\tilde{h}^n$), with the same amount of vanishing moments but of a higher order for the dual filter coefficients. Cohen Daubechies and Fauveau [57] designed a trigonometrical polynomial whose coefficients could be used in a convolution operation with h^n to produce $\tilde{h}^n$. The zeros of these particular coefficients are located near z=1 and produce a linear phase high pass filter. Fig. 6A1 shows the position of the zeros of the transfer function of this trigonometrical polynomial for 4 zeros (3^{rd} order) added to the 4 zeros from the transfer function of the binomial set of filter coefficients.

As an example, we will design a dual set of filter coefficients for B-spline biorthogonal wavelets with 3 vanishing moments. This will be done by using specially created functions for Matlab.

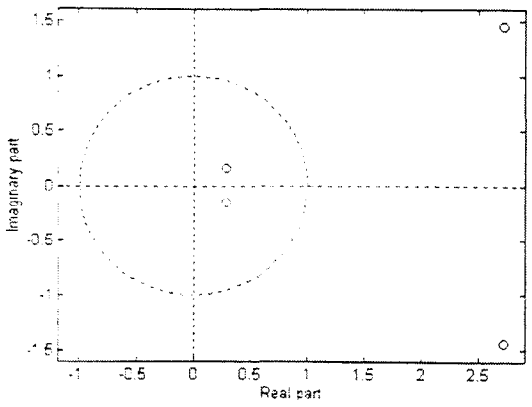


Fig.6ap.1. Zeros of transfer function of 3rd order trigonometric polynomial

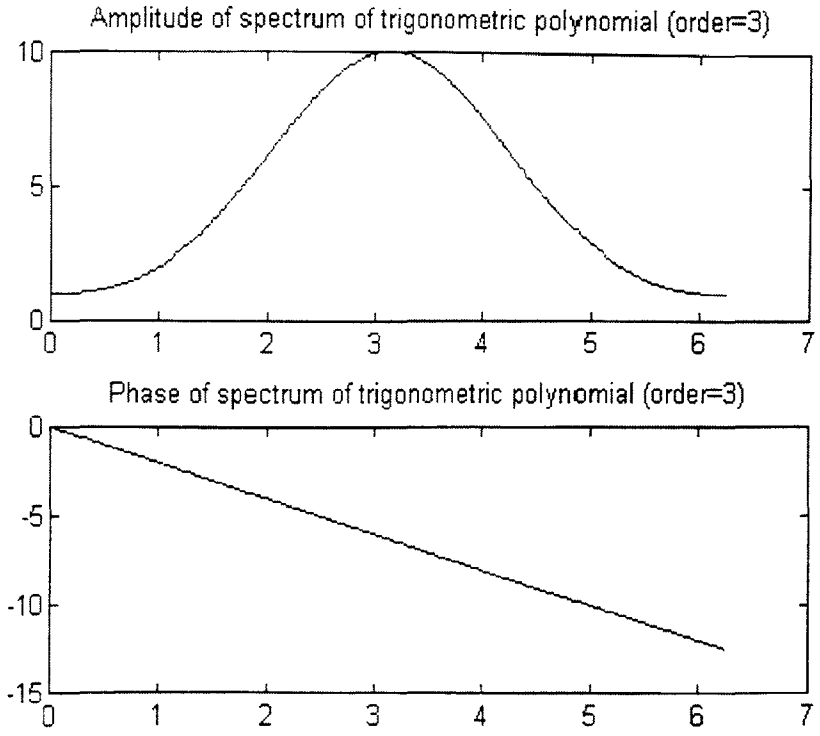


Fig.6ap.2 The above figures show respectively the amplitude and the phase of the filter whose transfer function will be multiplied with the transfer function of the low pass filter with the 3 vanishing moment.

Multiplying in the frequency domain results in convolution in the time domain . With some wavelet toolbox functions for Matlab we become :

```
p_3=trigpol(3)
```

p_3 = 0.3750 -2.2500 4.7500 -2.2500 0.3750

binomial = 1/2^3*(binewton(3))
binomial = 0.1250 0.3750 0.3750 0.1250

h_dual=conv(p_3,binomial)
h_dual = 0.0469 -0.1406 -0.1094 0.7031 0.7031 -0.1094 -0.1406 0.0469
= 3/64 -9/64 -7/64 45/64 45/64 -7/64 -9/64 3/64

The accompanying high pass filters can be very easily designed by multiplying the elements of h_dual with $(-1)^k$ (k=1...length(h_dual)). Wickerhausen [ ] built his tables of coefficient by using this concept.

Matrix-based design of biorthogonal filters

We will design a biorthogonal filter with Matlab tools and compare the results with the solution of a set of filter coefficient equations ensued from orthogonality and vanishing moment constraints.
The generation of the binomial filter coefficients, the coefficients of the trigonometric polynomial , the final convolution operation and the generation of high pass filter coefficients is assembled in one m-file : wspline.m

[h2_dual,g2_dual,h2,g2]=wspline(2,2)

h2_dual = -0.1768 0.3536 1.0607 0.3536 -0.1768
= √2 (-1/8 ¼ ¾ ¼ -1/8)

g2_dual = -0.3536 0.7071 -0.3536
= √2 (-¼ ½ -¼)

h2 = 0.3536 0.7071 0.3536
= √2 (¼ ½ ¼)

$$g2 = \begin{matrix} -0.1768 & -0.3536 & 1.0607 & -0.3536 & -0.1768 \end{matrix}$$

$$= \sqrt{2}\ \begin{pmatrix} -1/8 & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -1/8 \end{pmatrix}$$

Just like for normal orthogonal wavelets (see appendix 5b) the **matrix method** of describing wavelet transforms leads to an elegant way to design wavelet coefficients [68]. The products of the wavelet coefficients matrices for forward and backward transform must result in the unity matrix. To demonstrate this technique the simplest symmetric set with 3-5 coefficients lowpass: $[(h_0,h_1,h_0),(g_0,g_1,g_2,g_1,g_0)]$ and highpass: $[(-h_0,h_1,-h_0),(g_0,-g_1,g_2,-g_1,g_0)]$ is implemented in matrix M. Remark that the matrix construction is slightly different to the normal wavelet matrix. For instance , it starts from the centre wavelet coefficient value and the inverse matrix is not just the transpose of the original one. (For Mathcad notation sake we use g instead off $\tilde{h}$)

$$M := \begin{pmatrix} h_1 & h_0 & 0 & 0 & 0 & 0 & 0 & h_0 \\ 0 & h_0 & h_1 & h_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & h_0 & h_1 & h_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & h_0 & h_1 & h_0 \\ -g_1 & g_0 & 0 & 0 & 0 & g_0 & -g_1 & g_2 \\ -g_1 & g_2 & -g_1 & g_0 & 0 & 0 & 0 & g_0 \\ 0 & g_0 & -g_1 & g_2 & -g_1 & g_0 & 0 & 0 \\ 0 & 0 & 0 & g_0 & -g_1 & g_2 & -g_1 & g_0 \end{pmatrix} \quad M_{inv} := 2 \cdot \begin{pmatrix} g_2 & g_0 & 0 & g_0 & -h_0 & -h_0 & 0 & 0 \\ g_1 & g_1 & 0 & 0 & 0 & h_1 & 0 & 0 \\ g_0 & g_2 & g_0 & 0 & 0 & -h_0 & -h_0 & 0 \\ 0 & g_1 & g_1 & 0 & 0 & 0 & h_1 & 0 \\ 0 & g_0 & g_2 & g_0 & 0 & 0 & -h_0 & -h_0 \\ 0 & 0 & g_1 & g_1 & 0 & 0 & 0 & h_1 \\ g_0 & 0 & g_0 & g_2 & -h_0 & 0 & 0 & -h_0 \\ g_1 & 0 & 0 & g_1 & h_1 & 0 & 0 & 0 \end{pmatrix}$$

$$M_{inv}M = \begin{pmatrix} 2g_2h_1 - 4h_0g_1 & 0 & 2g_0h_1 + 2h_0g_1 & 0 & 0 & 0 & 2g_0h_1 + 2h_0g_1 & 0 \\ 0 & 2g_2h_1 - 4h_0g_1 & 0 & 2g_0h_1 - 2h_0g_1 & 0 & 0 & 0 & 2g_0h_1 + 2h_0g_1 \\ 2g_0h_1 - 2h_0g_1 & 0 & 2g_2h_1 + 4h_0g_1 & 0 & 2g_0h_1 + 2h_0g_1 & 0 & 0 & 0 \\ 0 & 2g_0h_1 + 2h_0g_1 & 0 & 2g_2h_1 + 4h_0g_1 & 0 & 2g_0h_1 + 2h_0g_1 & 0 & 0 \\ 0 & 0 & 2g_0h_1 + 2h_0g_1 & 0 & 2g_2h_1 + 4h_0g_1 & 0 & 2g_0h_1 + 2h_0g_1 & 0 \\ 0 & 0 & 0 & 2g_0h_1 + 2h_0g_1 & 0 & 2g_2h_1 + 4h_0g_1 & 0 & 2g_0h_1 + 2h_0g_1 \\ 2g_0h_1 + 2h_0g_1 & 0 & 0 & 0 & 2g_0h_1 + 2h_0g_1 & 0 & 2g_2h_1 + 4h_0g_1 & 0 \\ 0 & 2g_0h_1 + 2h_0g_1 & 0 & 0 & 0 & 2g_0h_1 + 2h_0g_1 & 0 & 2g_2h_1 + 4h_0g_1 \end{pmatrix}$$

By setting this matrix equal to the unity matrix 2 equation result . At least 3 other equations are needed to produce a solution. We add 2 equations requiring the sum of the high pass filter coefficients to be equal to 1. Another set of 2 equations comes

from the vanishing moments (0^{th} and 1^{st} order) of the wavelet. With some starting values the solution converges to a very elegant solution! The filter coefficients are rational numbers and at least the multiplication of the denominator value can be realised with shift operations. Combined with the lifting scheme on a VSP (see chapter 7), this must be very beneficial in real time video processing.

$h_0 := \frac{1}{3}$ $h_1 := \frac{1}{3}$ $g_0 := \frac{1}{5}$ $g_1 := \frac{1}{5}$ $g_2 := \frac{1}{5}$

Given

$2 \cdot g_2 \cdot h_1 + 4 \cdot h_0 \cdot g_1 = 1$

$2 \cdot g_0 \cdot h_1 + 2 \cdot h_0 \cdot g_1 = 0$

$-2 \cdot h_0 + h_1 = 0$

$2 \cdot g_0 - 2 \cdot g_1 - g_2 = 0$

$-g_1 + 2 \cdot g_2 - 3 \cdot g_1 + 4 \cdot g_0 = 0$

$2 \cdot h_0 + h_1 = 1$

$2 \cdot g_0 + 2 \cdot g_1 + g_2 = 1$

$v\_biorthogonal\_35 := find(h_0, h_1, g_0, g_1, g_2)$ $v\_biorthogonal\_35 = \begin{bmatrix} 0.25 \\ 0.5 \\ -0.125 \\ 0.25 \\ 0.75 \end{bmatrix}$

$$M_{inv} \cdot M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The matrix multiplication of M_{inv} and M proves the biorthogonality . Comparing this result with the Cohen Daubechies Feauveau technique, we come to an identical solution.

References

[1] Jonathan M. Blackledge ,Paul Lezeau . Solution to the Non-Stationary Deconvolution Problem. IMA Conference Proceedings in Image Processing : Mathematical Methods and Application, Oxford University Press.

[2] Olivier Rioul , Martin Vetterly . Wavelets and signal processing. IEEE SP Magazine October 91 p.14-38.

[3] Franz Hlawatch , G. Faye Boudreaux-Bartels . Linear and Quadratic Time-Frequency Signal Representation. IEEE SP Magazine April 92 p.21-67.

[4] Adhemar Bultheel . Learning to swim in a sea of wavelets. Bull. Belg. Math. Soc. 2 (1995) p.1-44 .

[5] Albert Cohen . Jelena Kovacevic. Wavelets : The Mathematical Background. Proceedings of the IEEE, Vol.84 No.4 p.514-522 April 1996 .

[6] John J. Benedetto , Michael W. Frazier. Wavelets. Mathematics and Applications . CRC Press 1994.

[7] Robert S. Strichartz. How to make wavelets. American Mathematical Monthly 100 June-July 93.

[8] Carl Taswell , Kevin C. McGill. Algorithm 735 : Wavelet Transform Algorithms for Finite-Duration Discrete-Time Signals. ACM Transaction on Mathematical Software , Vol. 20, No. 3, September 1994, p. 398-412.

[9] Stephane G. Mallat. Multiresolutional Approximations and Wavelet Orthogonal Bases of L^2 (R) Transactions of the American Mathematical Society Vol. 315 Number 1, September 1989.

[10] Wim Sweldens. The lifting Scheme : A New Philosophy in Biorthogonal Wavelet Constructions. Internal report Department of Computer Sciences KUL Belgium 1995.

[11] Bjorn Jawerth ,Wim Sweldens. An Overview of Wavelet Based Multiresolutional Analysis. Preprint on Internet site : sweldens@math.scarolina.edu.

[12] Wim Sweldens , Peter Schroder. Building Your Own Wavelets at Home. Preprint available via email see [11]

[13] Ingrid Daubechies. Orthogonal Bases of Compactly Supported Wavelets. Communications on Pure and Applied Mathematics, Vol. XLI p.909-996 1988.

[14] Gilbert Strang. The optimal coefficients in Daubechies wavelets. Physica D 60 p.239-244 1992.

[15] Nikolaj Hess-Nielsen , Mladen Victor Wickerhausen . Wavelets and Time - Frequency Analysis. Proceedings of the IEEE, Vol.84 No.4 April 1996.

[16] Kannan Ramchandran , Martin Vetterli , Cormac Herley. Wavelets, Subband Coding and Best Bases. Proceedings of the IEEE Vol.84 No.4 p.541-560 April 1996.

[17] Ingrid Daubechies . Where do Wavelets Come From ? - A Personal Point of View. Proceedings of the IEEE Vol.84 No.4 p.510-513 April 1996.

[18] Wim Sweldens . Wavelets : What Next? Internet publication see [11].

[19] A. Grossmann , J Morlet . Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape. Society for Industrial and Applied Mathematics 1984

[20] Michael Unser, Akram Aldroubi and Murray Eden . A Family of Polynomial Spline Wavelet Transforms. Signal Processing 30 (1993) 141-162.

[21] Michael Unser, Akram Aldroubi and Murray Eden . The L_2 Polynomial Spline Pyramid IEEE Transactions on Pattern Analysis and Machine Intelligence Vol. 15 N0. 4, April 1993.

[22] Tor Berger , Jan O. Stromberg. Exact Reconstruction Algorithms for the Discrete Wavelet Transform Using Spline Wavelets. Applied and Computational Harmonic Analysis 2, 392-397 (1995).

[23] Gauthier Lafruit , Tony Luckx, Jan Bormans, Ivo Bolsens, Jan Cornelis. Space-Filling Curves in Advanced Image Compression Applications. IMEC Belgium Internal paper 1996.

[24] Jean-Luc Starck, Fionn Murtach. Image Restoration with Noise Suppression using the Wavelet Transform. Astron. Astrophys. 288,342-248 (1994).

[25] Albert Bijaoui , Jean-Luc Starck , Fionn Murtagh. Restauration des Images Multi-Echelles par l'Algorithme a Trous Traitement du Signal Vol. 3 No11, 1994.

[26] Jean-Luc Starck , Fionn Murtach ,Albert Bijaoui .Multiresolution and Astronomical Image Processing. Communication at the "Conference on Astronomical Data Analysis and Software System" (ADASS 94) September 25-28, 1994.

[27] Jean-Luc Starck , Fionn Murtach , Albert Bijaoui . Multiresolution Support Applied to Image Filtering and Restoration. Submitted to "Graphical Models and Image Processing" September 1994.

[28] Jean-Luc Starck ,Albert Bijaoui. Bruno Lopez ,Christian Perrier. Image Reconstruction by the wavelet Transform Applied to Aperture Synthesis. Astronomy and Astrophysics 9.4.1993.

[29] Jean-Luc Starck , Albert Bijaoui . Filtering and deconvolution by the wavelet transform. Signal Processing 35 (1994) 195-211.

[30] J.G. Teti, Jr. , H.N. Kritikos. Synthetic Aperture Radar (SAR) Ocean Image Decomposition Using the Gabor Expansion. IEEE Transactions on Geoscience and Remote Sensing. Vol. 30 No.1, January1992.

[31] J.G. Teti, Jr., H.N. Kritikos . Synthetic Aperture Radar (SAR) Ocean Image Decomposition Using Wavelets. IEEE Transactions on Geoscience and Remote Sensing. Vol. 30 No.1, September1992.

[32] Avijit Chacraborty, David Okaya. Frequency-time decomposition of seismic data using wavelet-based methods. Geophysics, Vol.60 No.6 (November-December 1995) P1906-1916.

[33] Wavelets for Image Processing. IEEE Engineering in Medicine and Biology Magazine Vol.14 No. 5 September/October 1995.

[34] Jean-Pierre Antoine. Wavelet Analysis in Image Processing Signal Processing VI : Theories and Applications Elsevier 1992.

[35] Jean-Pierre Antoine, P. Carrette, R.Murenzi, B.Piette. Image analysis with Two-Dimensional Continuous Wavelet Transform . Signal Processing 31 1993 (241-272) .

[36] John G. Daugman. Complete Discrete 2-D Gabor Transform by Neural Networks for Image Analysis and Compression. IEEE Transactions on Acoustics, Speech, and Signal Processing Vol. 36. No.7 July 1988.

[37] Qinghua Zhang and Albert Benveniste . Wavelet Networks IEEE Transactions on Neural Networks, Vol. 3, No. 6, November 1992.

[38] Jacques Istas. Wavelet coefficients of a Gaussian process and applications Annales de l'Institut Henri Poincare Vol. 28 No. 4 p537-556 1992.

[39] Jonathan Berger , Ronald R. Coifman , Maxim J. Goldberg. Removing Noise from Music Using Local Trigonometric Bases and Wavelet Packets. Journal of Audio Engineering Society Vol. 42 No.10 October 1994.

[40] D. Gabor . Theory of Communications . J. Inst. Electr. Eng. (London) Vol. 93, No. III, pp. 429-457, 1946.

.

[41] Francis Decroos . Wiener based Wavelet Filtering with Video Signal Processors. MSc. Thesis supervised by HT, Department of Electrical Engineering , Katholieke Hogeschool Brugge-Oostende, 1996.

[42] Kris Dierkens Wavelets and Wavelet Packets based Compression on TMS320C80 .MSc. Thesis supervised by HT, Department of Electrical Engineering , Katholieke Hogeschool Brugge-Oostende, 1996.

[43] G. Essink ,E.A. de Kock. VSP Programming Tools User Guide, technical note, 1996.

[44] Alan V. Oppenheim. A Personal View on Education, IEEE Signal Processing Magazine , April 1992.

[45] R. Pasko, H. Tassignon, D.Durackova. Design of Object Oriented Subroutines in C++ for Digital Image Processing . Invited paper at the System Design Conference in Bratislava 1994.

[46] Vincent Pollet Advanced Methods for Object Recognition . MSc. Thesis supervised by HT, Department of Electrical Engineering . Katholieke Hogeschool Brugge-Oostende, 1996.

[47] R. Rousseau , H. Tassignon Computer Aided Learning of Fourier Transforms using Mathcad . Proceedings of SEFI conference Prague 1995.

[48] F. Decroos, K. Dierkens, V. Pollet, R. Rousseau, H. Tassignon, K. Verweyen. Decorrelation Techniques in Bibliometry Bibliometrical magazine, accepted to be published in 1997.

[49] A. Sinnaeve, H. Tassignon. Signal Averaged ECG. p.93-115. In 'Cardiac Pacing and Electrophysiology. A bridge to the 21st century.' Kluwer Academic Publishers 1994.

[50] H. Tassignon . Wiener filtering in Wavelet Space . Invited paper at the EDS96 conference in Brno 1996.

[51] K.A. Vissers, G. Essink, P.H.J. van Gerwen, P.J.M. Janssen, O. Popp, E. Riddersma, W.J.M. Smits, H.J.M. Veendrick. Architecture and programming of two generations video signal processors, Microprocessing and Microprogramming, no. 41, pp 373-390, 1995.

[52] K. Verweyen. From a real time video algorithm to an efficient ASIC solution, MSc. Thesis, Department of Electrical Engineering , Katholieke Hogeschool Brugge-Oostende, 1996.

[53] C. Davidson & D.Rock . Wavelets and HONN : Pix-Perfect Marriage , AI Expert 1995.

[54] Gabriel Fernandez, S. Periaswamy, Wim Sweldens. Liftpack, A Software Package for Wavelet Transforms using Lifting, SPIE 1996.

[55] W.H. Press, B.P. Flannery , S.A. Teukolsky , W.T. Vettering. Numerical Recipes. Cambridge University Press 2nd edition, 1993.

[56] Kenneth R. Castleman . Digital Image Processing Prentice Hall International 1996.

[57] Ingrid Daubechies . Ten Lectures on Wavelets SIAM Philadelphia 1992

[58] Barbara Burke Hubbard . The World According to Wavelets A.K.Peters Ltd., Wellesley, Ma, 1996.

[59] Jae S. Lim . Two-Dimensional Signal and Image Processing. Prentice Hall International. 1990.

[60] Timothy Masters Signals and Image Processing with Neural Networks. A C++ Sourcebook. John Wiley and Sons, Inc.1994.

[61] Yves Meyer. Wavelets , Algorithms & Applications Translated and Revised by Robert D. Ryan SIAM 1993.

[62] Alan V. Oppenheim, Ronald W. Schafer. Digital Signal Processing Prentice Hall 1975.

[63] Alan V. Oppenheim , Alan S. Willsky with Ian Young . Signals and Systems Prentice Hall International 1983.

[64] L. R. Rabiner , B. Gold . Theory and Applications of Digital Signal Processing Prentice Hall International 1975.

[65] B.Ruskai (ed). Wavelets and their applications , Jones and Bartlett Boston 1992.

[66] Gilbert G. Walter. Wavelets and Other Orthogonal Systems with Application. CRC Press 1994.

[67] M. Victor Wickerhausen. Adapted Wavelet analysis from Theory to Practice. A.K.Peters Ltd., Wellesley, Ma, 1994.

[68] H.J. Barnard . Image and Video Coding using Wavelet Decomposition, PhD thesis Technische Universiteit Delft The Netherlands (1994).

[69] Simon Haykin Adaptive Filter Theory Prentice Hall 1991